Nigel Smart (Ed.)

LNCS 4965

Advances in Cryptology – EUROCRYPT 2008

27th Annual International Conference on the Theory and Applications of Cryptographic Techniques Istanbul, Turkey, April 2008, Proceedings

INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH

iacr

Springer

Lecture Notes in Computer Science 4965

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

Nigel Smart (Ed.)

Advances in Cryptology – EUROCRYPT 2008

27th Annual International Conference on the Theory and Applications of Cryptographic Techniques
Istanbul, Turkey, April 13-17, 2008
Proceedings

Springer

Volume Editor

Nigel Smart
University of Bristol
Department of Computer Science
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK
E-mail: nigel@cs.bris.ac.uk

Library of Congress Control Number: 2008923745

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-78966-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-78966-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12251506 06/3180 5 4 3 2 1 0

Preface

These are the proceedings of Eurocrypt 2008, the 27th Annual IACR Eurocrypt Conference. The conference was sponsored by the International Association for Cryptologic Research ([illegible]), this year in cooperation with Tubitak ([illegible]).

The Eurocrypt 2008 Program Committee (PC) consisted of 28 members whose names are listed on the next page. There were 163 papers submitted to the conference and the PC chose 31 of them. Each paper was assigned to at least three PC members, who either handled it themselves or assigned it to an external referee. After the reviews were submitted, the committee deliberated both online for several weeks and finally in a face-to-face meeting held in Bristol. Papers were refereed anonymously, with PC papers having a minimum of five reviewers. All of our deliberations were aided by the [illegible] written and maintained by Shai Halevi. In addition to notification of the decision of the committee, authors received reviews; the default for any report given to the committee was that it should be available to the authors as well.

The committee decided to give the Best Paper Award to Ben Smith for his paper "Isogenies and the Discrete Logarithm Problem in Jacobians of Genus 3 Hyperelliptic Curves." The conference program also included two invited lectures: one by Andy Clark entitled "From Gamekeeping to Poaching - Information Forensics and Associated Challenges," and the other by Clifford Cocks on "The Growth and Development of Public Key Cryptography."

I wish to thank all the people who made the conference possible. First and foremost the authors who submitted their papers. The hard task of reading, commenting, debating and finally selecting the papers for the conference fell on the PC members. Without the hard work of the committee members and their respective sub-reviewers the whole process would be so much harder to organize. I thank Shai Halevi for handling the submissions and reviews server, and for also organizing the phone conference for those people unable to attend the Bristol PC meeting in person. I am also grateful to members of the IACR board, and previous PC Chairs, who shared their invaluable advice with me and to A. Murat Apohan and the rest of the local Organizing Committee.

Finally, I would like to say that it has been a great honor to be PC Chair for Eurocrypt 2008 and I only hope the readers of the following papers obtain as much enjoyment as I did in selecting them.

February 2008 Nigel Smart

EuroCrypt 2008

Istanbul, Turkey, April 13–17

Sponsored by [illegible]

Organized in cooperation with [illegible]

General Chair

A. Murat Apohan

Program Chair

Nigel Smart, University of Bristol

Program Committee

Sasha Boldyreva	Georgia Institute of Technology
Orr Dunkelman	K.U. Leuven
Serge Fehr	CWI
Eiichiro Fujisaki	NTT
Shai Halevi	IBM
Antoine Joux	DGA and Université de Versailles
Marc Joye	Thomson R&D
Jonathan Katz	University of Maryland
Lars R. Knudsen	DTU Mathematics
Yoshi Kohno	University of Washington
Arjen Lenstra	EPFL and Alcatel-Lucent Bell Laboratories
Stefan Lucks	B.-U. Weimar
Anna Lysyanskaya	Brown University
Wenbo Mao	EMC Research China
Alfred Menezes	Univeristy of Waterloo
Atsuko Miyaji	JAIST
David Naccache	ENS
Gregory Neven	K.U. Leuven
Phong Nguyen	CNRS and ENS
Kenny Paterson	Royal Holloway, University of London
Matt Robshaw	Orange Labs
Ahmad-Reza Sadeghi	Ruhr University Bochum
Rei Safavi-Naini	University of Calgary

Berry Schoenmakers	T.U. Eindhoven
Hovav Shacham	UC San Diego
Brent Waters	SRI
Stefan Wolf	ETH Zurich
Moti Yung	Google

External Reviewers

Michel Abdalla
Masayuki Abe
Omran Ahmadi
Elena Andreeva
Kazumaro Aoki
Frederik Armknecht
Giuseppe Ateniese
Nuttapong Attrapadung
Roberto Avanzi
Mira Belenkiy
Côme Berbain
Guido Blady
Andrey Bogdanov
Colin Boyd
Matt Campagna
Sébastien Canard
Ran Canetti
David Cash
Dario Catalano
Nishanth Chandran
Melissa Chase
Sanjit Chatterjee
Benoît Chevallier-Mames
Carlos Cid
Scott Contini
Jason Crampton
Ivan Damgård
Cécile Delerablée
Claus Diem
Yevgeniy Dodis
Dror Eiger
Jean-Charles Faugère
Marc Fischlin
Matthias Fitzi
Ewan Fleischmann
Pierre-Alain Fouque
Georg Fuchsbauer
Eiichiro Fujisaki
Jun Furukawa
Steven Galbraith
Nicolas Gama
Pierrick Gaudry
Praveen Gauravaram
Rosario Gennaro
Henri Gilbert
Vanessa Gratzer
Robbert de Haan
Goichiro Hanaoka
Darrel Hankerson
Guillaume Hanrot
Aram Harrow
Johan Håstad
Florian Hess
Jason Hinek
Martin Hirt
Dennis Hofheinz
Susan Hohenberger
Nick Howgrave-Graham
Yuval Ishai
Mitsugu Iwamoto
Stas Jarecki
Shaoquan Jiang
Ellen Jochemsz
Pascal Junod
Ulrich Kühn
Marcelo Kaihara
Alexandre Karlov
Mohamed Karroumi
Shahram Khazaei
Hiroaki Kikuchi
Eike Kiltz
Mehmet Kiraz
Takashi Kitagawa
Kazukuni Kobara

Hiroki Koga
Markulf Kohlweiss
Vlad Kolesnikov
Yuichi Komano
Matthias Krause
Hugo Krawczyk
Virendra Kumar
Alptekin Küpçü
Kaoru Kurosawa
Tanja Lange
Kristin Lauter
Stephane Lemieux
Gaetan Leurent
Eric Levieil
Jun Li
Benoît Libert
Hans Loehr
Bodo Möller
Phil MacKenzie
Subhomay Maitra
Mark Manulis
Sandra Marcello
Keith Martin
Ueli Maurer
Alexander May
Daniele Micciancio
Lorenz Minder
Toshiyuki Miyazawa
Sean Murphy
Toru Nakanishi
Jesper Buus Nielsen
Svetla Nikova
Ryo Nishimaki
Masayuki Numao
Wakaha Ogata
Koji Okada
Tatsuaki Okamoto
Elisabeth Oswald
Roger Oyono
Carles Padró
Dan Page
Pascal Paillier
Adriana Palacio
Omkant Pandey
Rafael Pass
Chris Peikert
Olivier Pereira
Ludovic Perret
Krzysztof Pietrzak
David Pointcheval
Marcin Poturalski
Manoj Prabhakaran
Tal Rabin
Vaibhav Rajan
Vibhor Rastogi
Dominik Raub
Eric Rescorla
Reza Reyhanitabar
Leonid Reyzin
Thomas Ristenpart
Louis Salvail
Jorge Villar Santos
Yu Sasaki
Nitesh Saxena
Rüdiger Schack
Werner Schindler
Yannick Seurin
Siamak Shahandashti
Emily Shen
Junji Shikata
Jamshid Shokrollahi
Thomas Shrimpton
Andrey Sidorenko
Adam Smith
Masakazu Soshi
Igor Spharlinski
Michal Sramka
Martijn Stam
Heiko Stamer
François-Xavier Standaert
Dirk Stegemann
Willy Susilo
Koutarou Suzuki
Keisuke Tanaka
Emin Tatli
Yael Tauman-Kalai
Stefano Tessaro
Soeren Steffen Thomsen
Tomas Toft
Frederik Vercauteren

Damien Vergnaud
José Villegas
Marcus Wagner
Bogdan Warinschi
Hoeteck Wee
Daniel Wichs
Andreas Winter
Christopher Wolf
David Woodruff
Jürg Wullschleger
Brecht Wyseur
Keita Xagawa
Go Yamamoto
Kan Yasuda
Kazuki Yoneyama
Rui Zhang
Yunlei Zhao
Sebastien Zimmer

Table of Contents

A Practical Attack on KeeLoq*

Sebastiaan Indesteege[1,**], Nathan Keller[2,***], Orr Dunkelman[1], Eli Biham[3], and Bart Preneel[1]

[1] Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven. Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
{sebastiaan.indesteege,orr.dunkelman,bart.preneel}@esat.kuleuven.be
[2] Einstein Institute of Mathematics, Hebrew University. Jerusalem 91904, Israel
nkeller@math.huji.ac.il
[3] Computer Science Department, Technion. Haifa 32000, Israel
biham@cs.technion.ac.il

Abstract. KeeLoq is a lightweight block cipher with a 32-bit block size and a 64-bit key. Despite its short key size, it is widely used in remote keyless entry systems and other wireless authentication applications. For example, authentication protocols based on KeeLoq are supposedly used by various car manufacturers in anti-theft mechanisms. This paper presents a practical key recovery attack against KeeLoq that requires 2^{16} known plaintexts and has a time complexity of $2^{44.5}$ KeeLoq encryptions. It is based on the slide attack and a novel approach to meet-in-the-middle attacks. The fully implemented attack requires 65 minutes to obtain the required data and 7.8 days of calculations on 64 CPU cores. A variant which requires 2^{16} chosen plaintexts needs only 3.4 days on 64 CPU cores. Using only 10 000 euro, an attacker can purchase a cluster of 50 dual core computers that will find the secret key in about two days. We investigated the way KeeLoq is intended to be used in practice and conclude that our attack can be used to subvert the security of real systems. An attacker can acquire chosen plaintexts in practice, and one of the two suggested key derivation schemes for KeeLoq allows to recover the master secret from a single key.

Keywords: KeeLoq, cryptanalysis, block ciphers, slide attacks, meet-in-the-middle attacks.

1 Introduction

The KeeLoq technology [13] by Microchip Technology Inc. includes the KeeLoq block cipher and several authentication protocols built on top of it. The KeeLoq

* This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.
** F.W.O. Research Assistant, Fund for Scientific Research — Flanders (Belgium).
*** This author is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 1–18, 2008.

block cipher allows for very low cost and power efficient hardware implementations. This property has undoubtedly contributed to the popularity of the cipher in various wireless authentication applications. For example, multiple car manufacturers supposedly use, or have used KeeLoq to protect their cars against theft [5,6,7,9,17].[1]

Despite its design in the 80's, the first cryptanalysis of KeeLoq was only published by Bogdanov [5] in February 2007. This attack is based on the slide technique and a linear approximation of the non-linear Boolean function used in KeeLoq. The attack has a time complexity of 2^{52} KeeLoq encryptions and requires 16 GB of storage. It also requires the entire codebook, i.e., 2^{32} known plaintexts.

Courtois et al. apply algebraic techniques to cryptanalyse KeeLoq [7,9]. Although a direct algebraic attack fails for the full cipher, they reported various successful slide-algebraic attacks. For example, they claim that an algebraic attack can recover the key when given a slid pair in 2.9 seconds on average. As there is no way to ensure or identify the existence of a slid pair in the data sample, the attack is simply repeated 2^{32} times, once for each pair generated from 2^{16} known plaintexts. They also described attacks requiring the entire codebook, which exploit certain assumptions with respect to fixed points of the internal state. The fastest of these requires 2^{27} KeeLoq encryptions and has an estimated success probability of 44% [9].

In [6], Bogdanov published an updated version of his attack. A refined complexity analysis yields a slightly smaller time complexity, i.e., $2^{50.6}$ KeeLoq encryptions while still requiring the entire codebook. This paper also includes an improvement using the work of Courtois et al. [7] on the cycle structure of the cipher. We note that the time complexity of the attack using the cycle structure given in [6] is based on an assumption from an earlier version of [7], that a random word can be read from 16 GB of memory with a latency of only 1 clock cycle. This is very unrealistic in a real machine, so the actual time complexity is probably much higher. In a later version of [7], this assumption on the memory latency was changed to be 16 clock cycles.

Our practical attack is based on the slide attack as well. However, unlike other attacks, we combine it with a novel meet-in-the-middle attack. The optimised version of the attack uses 2^{16} known plaintexts and has a time complexity of $2^{44.5}$ KeeLoq encryptions. We have implemented our attack and the total running time is roughly 500 days. As the attack is fully parallelizable, given x CPU cores, the total running time is only $500/x$ days. A variant which requires 2^{16} chosen plaintexts needs only $218/x$ days on x CPU cores. For example, for 10 000 euro, one can obtain 50 dual core computers, which will take about two days to find the key. Another, probably even cheaper, though illegal option would be to rent a botnet to carry out the computations.

KeeLoq is used in two protocols, the "Code Hopping" and the "Identify Friend or Foe (IFF)" protocol. In practice, the latter protocol, a simple challenge response protocol, is the most interesting target to acquire the data that is necessary to

[1] We verified these claims to the best of our ability, however, no car manufacturer seems eager to publically disclose which algorithms are used.

Table 1. An overview of the known attacks on KeeLoq

Attack Type	Complexity			Reference
	Data	Time	Memory	
Time-Memory Trade-Off	2 CP	$2^{42.7}$	≈ 100 TB	[11]
Slide/Algebraic	2^{16} KP	$2^{65.4}$	?	[7,9]
Slide/Algebraic	2^{16} KP	$2^{51.4}$	?	[7,9]
Slide/Guess-and-Determine	2^{32} KP	2^{52}	16 GB	[5]
Slide/Guess-and-Determine	2^{32} KP	$2^{50.6}$	16 GB	[6]
Slide/Cycle Structure	2^{32} KP	$2^{39.4}$	16.5 GB	[7]
Slide/Cycle/Guess-and-Det.[a]	2^{32} KP	(2^{37})	16.5 GB	[6]
Slide/Fixed Points	2^{32} KP	2^{27}	> 16 GB	[9]
Slide/Meet-in-the-Middle	2^{16} KP	$2^{45.0}$	≈ 2 MB	Sect. 3.3
Slide/Meet-in-the-Middle	2^{16} KP	$2^{44.5}$	≈ 3 MB	Sect. 3.4
Slide/Meet-in-the-Middle	2^{16} CP	$2^{44.5}$	≈ 2 MB	Sect. 3.5
Time-Memory-Data Trade-Off	68 CP, 34 RK	$2^{39.3}$	≈ 10 TB	[2]
Related Key	66 CP, 34 $\text{RK}^{\ggg}$	negligible	negligible	Sect. A.1
Related Key	512 CP, 2 $\text{RK}^{\ggg}$	2^{32}	negligible	Sect. A.1
Related Key/Slide/MitM	2^{17} CP, 2 RK	$2^{41.9}$	≈ 16 MB	Sect. A.2

Time complexities are expressed in full KeeLoq encryptions (528 rounds).
KP: known plaintexts; CP: chosen plaintexts
$\text{RK}^{\ggg}$: related keys (by rotation); RK : related keys (flip LSB)

[a] The time complexity for this attack is based on very unrealistic memory latency assumptions and hence will be much higher in practice.

mount the attack. Because the challenges are not authenticated in any way, an attacker can obtain as many chosen plaintext/ciphertext pairs as needed from a transponder (e.g., a car key) implementing this protocol. Depending on the transponder, it takes 65 or 98 minutes to gather 2^{16} plaintext/ciphertext pairs.

Finally, as was previously noted by Bogdanov [6], we show that one of the two suggested key derivation algorithms is blatantly flawed, as it allows an attacker to reconstruct many secret keys once a single secret key has been exposed.

Given that KeeLoq is a cipher that is widely used in practice, side-channel analysis may also be a viable option for attacking chips that implement KeeLoq. However, we do not consider this type of attack in this paper. One could also attack the “Identify Friend or Foe (IFF)” protocol itself. For instance, as the responses are only 32 bits long, one could mount the birthday attack using 2^{16} known challenge/response pairs. This would not recover the secret key, thus posing less of a threat to the overall security of the system.

Table 1 presents an overview of the known attacks on KeeLoq, including ours. In order to make comparisons possible, we have converted all time complexities to the number of KeeLoq encryptions needed for the attack.[2]

[2] We list slightly better complexities for the attacks from [7,9] because we used a more realistic conversion factor from CPU clocks to KeeLoq rounds (i.e., 12 rather than 4 CPU cycles per KeeLoq round).

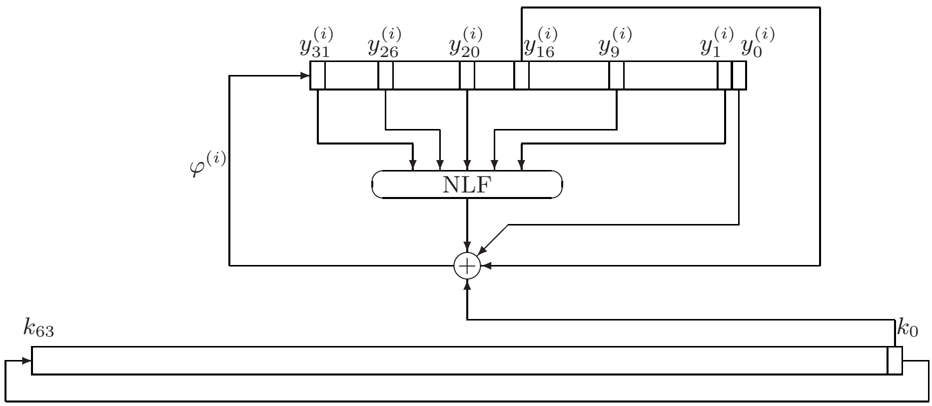


Fig. 1. The i-th KeeLoq encryption cycle

The structure of this paper is as follows. In Sect. 2, we describe the KeeLoq block cipher and how it is intended to be used in practice. Our attacks are described in Sect. 3. In Sect. 4 we discuss our experimental results and in Sect. 5 we show the relevance of our attacks in practice. Finally, in Sect. 6 we conclude. In Appendix A, we explore some related key attacks on KeeLoq that are more of theoretical interest.

2 Description and Usage of KeeLoq

2.1 The KeeLoq Block Cipher

The KeeLoq block cipher has a 32-bit block size and a 64-bit key. It consists of 528 identical rounds each using one bit of the key. A round is equivalent to an iteration of a non-linear feedback shift register (NLFSR), as shown in Fig. 1.

More specifically, let $Y^{(i)} = (y_{31}^{(i)}, \ldots, y_0^{(i)}) \in \{0,1\}^{32}$ be the input to round i $(0 \leq i < 528)$ and let $K = (k_{63}, \ldots, k_0) \in \{0,1\}^{64}$ be the key. The input to round 0 is the plaintext: $Y^{(0)} = P$. The ciphertext is the output after 528 rounds: $C = Y^{(528)}$. The round function can be described as follows (see Fig. 1):

$$\begin{aligned} \varphi^{(i)} &= \mathrm{NLF}\left(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_{9}^{(i)}, y_{1}^{(i)}\right) \oplus y_{16}^{(i)} \oplus y_{0}^{(i)} \oplus k_{i \bmod 64} \ , \\ Y^{(i+1)} &= (\varphi^{(i)}, y_{31}^{(i)}, \ldots, y_{1}^{(i)}) \ . \end{aligned} \tag{1}$$

The non-linear function NLF is a Boolean function of 5 variables with output vector $3A5C742E_x$ — i.e., NLF(i) is the i-th bit of this hexadecimal constant, where bit 0 is the least significant bit. We can also represent the non-linear function in its algebraic normal form (ANF):

$$\begin{aligned} \mathrm{NLF}(x_4, x_3, x_2, x_1, x_0) = \ & x_4x_3x_2 \oplus x_4x_3x_1 \oplus x_4x_2x_0 \oplus x_4x_1x_0 \oplus \\ & x_4x_2 \oplus x_4x_0 \oplus x_3x_2 \oplus x_3x_0 \oplus x_2x_1 \oplus x_1x_0 \oplus \\ & x_1 \oplus x_0 \ . \end{aligned} \tag{2}$$

Decryption uses the inverse round function, where i now ranges from 528 down to 1.

$$\begin{aligned} \theta^{(i)} &= \mathrm{NLF}\left(y_{30}^{(i)}, y_{25}^{(i)}, y_{19}^{(i)}, y_{8}^{(i)}, y_{0}^{(i)}\right) \oplus y_{15}^{(i)} \oplus y_{31}^{(i)} \oplus k_{i-1 \bmod 64} \ , \\ Y^{(i-1)} &= (y_{30}^{(i)}, \ldots, y_{0}^{(i)}, \theta^{(i)}) \ . \end{aligned} \tag{3}$$

There used to be some ambiguity about the correct position of the taps. Our description agrees with the "official" documentation [5,6,9,15]. Additionally, we have used test vectors generated by an actual HSC410 chip [14], manufactured by Microchip Inc., to verify that our description and implementation of KeeLoq are indeed correct. Finally, we note that our attacks are unaffected by this difference.

2.2 Protocols Built on KeeLoq

A device like the HCS410 by Microchip Technology Inc. [14] supports two authentication protocols based on KeeLoq: "KeeLoq Hopping Codes" and "KeeLoq Identify Friend or Foe (IFF)". The former uses a 16-bit secret counter, synchronised between both parties. In order to authenticate, the encoder (e.g., a car key) increments the counter and sends the encrypted counter value to the decoder (e.g., the car), which verifies if the received ciphertext is correct. In practice, this protocol would be initiated by a button press of the car owner.

The second protocol, "KeeLoq Identify Friend or Foe (IFF)" [14], is a simple challenge response protocol. The decoder (e.g., the car) sends a 32-bit challenge. The transponder (e.g., the car key) uses the challenge as a plaintext, encrypts it with the KeeLoq block cipher[3] under the shared secret key, and replies with the ciphertext. This protocol is executed without any user interaction whenever the transponder receives power and an activation signal via inductive coupling from a nearby decoder. Hence, no battery or button presses are required. It could for instance be used in vehicle immobilisers by placing the decoder near the ignition. Inserting the car key in the ignition would place the transponder within range of the decoder. The latter would then activate the transponder and execute the protocol, all completely transparent to the user. The car would then either disarm the immobiliser or activate the alarm, depending on whether the authentication was successful.

Of course both protocols can be used together in a single device, thereby saving costs. For example, the HCS410 chip [14] supports this combined mode of operation, possibly using the same secret key for both protocols, depending on the configuration options used.

3 Our Attacks on KeeLoq

This section describes our attacks on KeeLoq. We combine a slide attack with a novel meet-in-the-middle approach to recover the key from a slid pair. First we

[3] This corresponds to what is called the "HOP algorithm" in [14]. The other option, the so-called "IFF algorithm", uses a reduced version of KeeLoq with 272 rounds instead of 528. Our attacks are also applicable to this variant, without any change.

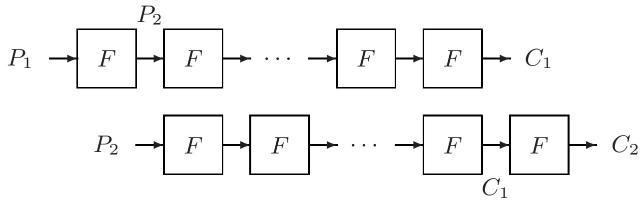


Fig. 2. A typical slide attack

explain some preliminaries that are used in the attacks. Then we proceed to the description of the attack scenario using known plaintexts and a generalisation thereof. Finally, we show how chosen plaintexts can be used to improve the attack.

3.1 The Slide Property

Slide attacks were introduced by Biryukov and Wagner [3] in 1999. The typical candidate for a slide attack is a block cipher consisting of a potentially very large number of iterations of an identical key dependent permutation F. In other words, the subkeys are repeated and therefore the susceptible cipher can be written as

$$C = \underbrace{F\left(F\left(\ldots F(P)\right)\right)}_{r} = F^r(P) \ . \tag{4}$$

This permutation does not necessarily have to coincide with the rounds of the cipher, i.e., F might combine several rounds of the cipher.

A slide attack aims at exploiting such a self-similar structure to reduce the strength of the entire cipher to the strength of F. Thus, it is independent of the number of rounds of the cipher. To accomplish this, a so-called [illegible] is needed. This is a pair of plaintexts that satisfies the slide property

$$P_2 = F(P_1) \ . \tag{5}$$

We depict such a slid pair in Fig. 2. For a slid pair, the corresponding ciphertexts also satisfy the slide property, i.e., $C_2 = F(C_1)$. By repeatedly encrypting this slid pair, we can generate as many slid pairs as needed [4,10]. As each slid pair gives us a pair of corresponding inputs and outputs of the key dependent permutation F, it can be used to mount an attack against F.

KeeLoq has 528 identical rounds, each using one bit of the 64-bit key. After 64 rounds the key is repeated. So in the case of KeeLoq, we combine 64 rounds into F. However, because the number of rounds in the cipher is not an integer multiple of 64, a straightforward slid attack is not possible. A solution to this problem is to guess the 16 least significant bits of the key and use this to strip off the final 16 rounds. Then, a slide attack can be applied to the remaining 512 rounds [5,7,9].

In order to get a slid pair, 2^{16} known plaintexts are used. As the block size of KeeLoq is 32 bits, we expect that a random set of 2^{16} plaintexts contains a slid pair due to the birthday paradox.[4] Determining which pair is a slid pair is done by the attack itself. Simply put, the attack is attempted with every pair. If it succeeds, the pair is a slid pair, otherwise it is not.

3.2 Determining Key Bits

If two intermediate states of the KeeLoq cipher, separated by 32 rounds (or less) are known, all the key bits used in these rounds can easily be recovered. This was first described by Bogdanov [5], who refers to it as the "linear step" of his attack.

Let $Y^{(i)} = (y_{31}^{(i)}, \ldots, y_0^{(i)})$ and $Y^{(i+t)} = (y_{31}^{(i+t)}, \ldots, y_0^{(i+t)})$ be the two known states; $t \leq 32$. If we encrypt $Y^{(i)}$ by one round, the newly generated bit is

$$\varphi^{(i)} = \mathrm{NLF}\left(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_9^{(i)}, y_1^{(i)}\right) \oplus y_{16}^{(i)} \oplus y_0^{(i)} \oplus k_{i \bmod 64} \ . \tag{6}$$

Because of the non-linear feedback shift register structure of the round function and since $t \leq 32$, the bit $\varphi^{(i)}$ is equal to $y_{32-t}^{(i+t)}$, which is one of the bits of $Y^{(i+t)}$ and thus known. Hence

$$k_{i \bmod 64} = \mathrm{NLF}\left(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_9^{(i)}, y_1^{(i)}\right) \oplus y_{16}^{(i)} \oplus y_0^{(i)} \oplus y_{32-t}^{(i+t)} \ . \tag{7}$$

By repeating this t times, all t key bits can be recovered. The amount of computations that need to be carried out is equivalent to t rounds of KeeLoq. This simple step will prove to be very useful in our attack.

3.3 Basic Attack Scenario

We now describe the basic attack scenario, which uses 2^{16} known plaintexts. For clarity, the notation used is shown in Fig. 3 and a pseudocode overview is given in Fig. 4. We denote 16 rounds of KeeLoq by $g_{\hat{k}}$, where $\hat{k}$ denotes the 16 key bits used in these rounds. The 64-bit key k is split into four equal parts: $k = (\hat{k}_3, \hat{k}_2, \hat{k}_1, \hat{k}_0)$, where $\hat{k}_0$ contains the 16 least significant key bits.

As already mentioned in Sect. 3.1, the first step of the attack is to guess $\hat{k}_0$ — the 16 least significant bits of the key. This enables us to partially encrypt each of the 2^{16} plaintexts by 16 rounds (P_i to X_i) and partially decrypt each of the 2^{16} ciphertexts by 16 rounds (C_j to Y_j).

Encrypting X_i by 16 more rounds yields $X_i^\star$. Similarly, decrypting P_j by 16 rounds yields $P_j^\star$ (see Fig. 3). We denote the 16 most significant bits of $X_i^\star$ by $\overline{X_i^\star}$, and the 16 least significant bits of $P_j^\star$ by $\underline{P_j^\star}$. Note that, because $X_i^\star$ and $P_j^\star$ are separated by 16 rounds, it holds that $\overline{X_i^\star} = \underline{P_j^\star}$, provided that P_i and P_j form a slid pair. This is due to the structure of the cipher.

[4] The probability that a set of 2^{16} random plaintexts contains at least one slid pair is $1 - \left(1 - 2^{\ 32}\right)^{2^{32}} \approx 0.63$. Hence, the attack has a success probability of about 63%. With not much higher data complexity, higher success rates can be achieved.

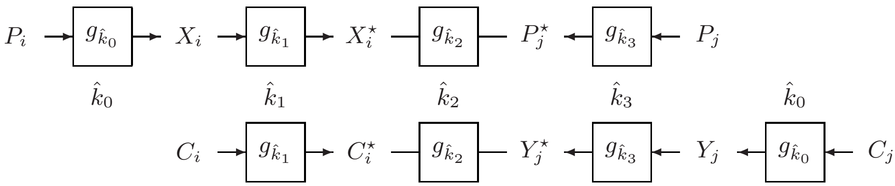


Fig. 3. The notation used in the attack

for all $\hat{k}_0 \in \{0,1\}^{16}$ **do**
  for all plaintexts $P_i, 0 \leq i < 2^{16}$ **do**
    Partially encrypt P_i to X_i.
    Partially decrypt C_i to Y_i.
  for all $\underline{P_j^\star} \in \{0,1\}^{16}$ **do**
    for all plaintexts $P_j, 0 \leq j < 2^{16}$ **do**
      Determine the key bits $\hat{k}_3$.
      Partially decrypt Y_j to $Y_j^\star$.
      Save the tuple $\left\langle P_j^\star, Y_i^\star, \hat{k}_3 \right\rangle$ in a table.
    for all plaintexts $P_i, 0 \leq i < 2^{16}$ **do**
      Determine the key bits $\hat{k}_1$.
      Partially encrypt C_i to $C_i^\star$.
      for all collisions $\overline{C_i^\star} = \underline{Y_j^\star}$ in the table **do**
        Determine the key bits $\hat{k}_2$ from $X_i^\star$ and $P_j^\star$.
        Determine the key bits $\hat{k}_2$ from $C_i^\star$ and $Y_j^\star$.
        if $\hat{k}_2 = \hat{k}_2$ **then**
          Encrypt 2 known plaintexts with the key $k = (\hat{k}_3, \hat{k}_2, \hat{k}_1, \hat{k}_0)$.
          if the correct ciphertexts are found **then**
            return success (the key is k)
return failure (i.e., there was no slid pair)

Fig. 4. The attack algorithm

The next step in the attack is to apply a meet-in-the-middle approach. We guess the 16-bit value $\underline{P_j^\star}$. For each plaintext P_j we can then determine $\hat{k}_3$ using the algorithm described in Sect. 3.2. Indeed, as the other bits of $P_j^\star$ are determined by P_j, we know all of $P_j^\star$ when given the plaintext. There is always exactly one solution per plaintext. Using this part of the key, we can now partially decrypt Y_j to $Y_j^\star$. This result is saved in a hash table indexed by the 16-bit value $\underline{Y_j^\star}$. Each record in the hash table holds a tuple consisting of $P_j^\star$, $Y_j^\star$ and the 16 key bits $\hat{k}_3$.

Now we do something similar from the other side. For each plaintext we use the algorithm from Sect. 3.2 to determine $\hat{k}_1$. Again this can be done because we know all of $X_i^\star$, and there is exactly one solution per plaintext. Knowing $\hat{k}_1$, we partially encrypt C_i to $C_i^\star$.

Note that if P_i and P_j are indeed a slid pair their partial encryptions and decryptions (under the correct key) must "meet in the middle". More specifically, it must hold that $\overline{C_i^\star} = \underline{Y_j^\star}$. So, we look for a record in the hash table for which such a collision occurs. Because the hash table is indexed by $\underline{Y_j^\star}$ this can be done very efficiently. A slid pair produces a collision, provided the guesses for $\hat{k}_0$ and $\underline{P_j^\star}$ are correct. Therefore, we are guaranteed that all slid pairs are found at some point. Of course, a collision does not guarantee that the pair is actually a slid pair.

Finally, we check each candidate slid pair found. We determine the remaining key bits $\hat{k}_2$ from $X_i^\star$ and $P_j^\star$ and similarly $\hat{k}_2'$ from $C_i^\star$ and $Y_j^\star$. If $\hat{k}_2$ and $\hat{k}_2'$ are not equal, the candidate pair is not a slid pair. Note that we can determine the key bits one by one and stop as soon as there is a disagreement. This slightly reduces the complexity of the attack.

If $\hat{k}_2 = \hat{k}_2'$, we have found a pair of plaintexts and a key with the property that encrypting P_i by 64 rounds gives P_j and encrypting C_i by 64 rounds gives C_j. This is what is expected from a slid pair. It is however possible that the recovered key is not the correct key, so we can verify it by a trial encryption of one of the known plaintexts. Even if a wrong key is suggested during the attack, and discarded by the trial encryption, we are still guaranteed to find the correct key eventually, provided there is at least one slid pair among the given plaintexts.

Complexity Analysis. Using one round of KeeLoq as a unit, the time complexity of the attack can be expressed as

$$2^{16}\left(32 \cdot 2^{16} + 2^{16}\left(32 \cdot 2^{16} + 2^{16}\left(32 + N_{\text{coll}} \cdot V\right)\right)\right) \quad , \tag{8}$$

when N_{coll} denotes the expected number of collisions for a single guess of $\hat{k}_0$, $\underline{P_j^\star}$ and a given plaintext P_i, and V denotes the average cost of verifying one collision, i.e., checking if it leads to a candidate key and if this key is correct. This follows directly from the description of the attack. As the hash table has 2^{16} entries and a collision is equivalent to a 16-bit condition, $N_{\text{coll}} = 1$. In the verification step, we can determine one bit at a time and stop as soon as there is a disagreement, which happens with probability 1/2. Only when there is no disagreement after 16 key bits, we do two full trial encryptions to check the recovered key. Of course the second trial encryption is only useful if the first one gave the expected result. Hence, due to this early abort technique, the average cost of verifying one collision is

$$V = 2 \cdot \sum_{i=0}^{15} 2^{-i} + 2^{-16} \cdot \left(528 + 528 \cdot 2^{-32}\right) \approx 4 \quad . \tag{9}$$

Thus the overall complexity of the attack is $2^{54.0}$ KeeLoq rounds, which amounts to $2^{45.0}$ full KeeLoq encryptions.

As mentioned before, the data complexity of the attack is 2^{16} known plaintexts. The storage requirements are very modest. The attack stores the plaintext/ciphertext pairs, 2^{16} values for X_i and Y_i, and a hash table with 2^{16} records of 80 bits each. This amounts to a bit over 2 MB of RAM.

3.4 A Generalisation of the Attack

The attack presented in the previous section can be generalised by varying the number of rounds to partially encrypt/decrypt in each step of the attack. We denote by t_p the number of rounds to partially encrypt from the plaintext side (left on Fig. 3) and by t_c the number of rounds to partially decrypt from the ciphertext side (right on Fig. 3). More specifically, encrypting X_i by t_p rounds yields $X_i^\star$, encrypting C_i by t_p rounds yields $C_i^\star$. On the ciphertext side, $P_j^\star$ is obtained by decrypting P_j by t_c rounds and $Y_j^\star$ by decrypting Y_j by t_c rounds. Also, the partial keys $\hat{k}_0$ through $\hat{k}_3$ are adapted accordingly to contain the appropriate key bits.

Let t_o denote the number of bits that, provided P_i and P_j form a slid pair, overlap between $X_i^\star$ and $P_j^\star$. As $X_i^\star$ and $P_j^\star$ are separated by $48 - t_p - t_c$ rounds, it holds that $t_o = 32 - (48 - t_p - t_c) = t_p + t_c - 16$. The t_o least significant bits of $P_j^\star$ are denoted by $\underline{P_j^\star}$ and the t_o most significant bits of $X_i^\star$ are denoted by $\overline{X_i^\star}$.

Depending on the choices for the parameters t_p and t_c, the attack scenario has to be modified slightly. If $t_c < t_o$, not all plaintexts necessarily yield a solution for a given $\underline{P_j^\star}$ when determining $\hat{k}_3 = (k_{63}, \ldots, k_{64-t_c})$ because $t_o - t_c$ of the guessed bits overlap with plaintext bits. Similarly, if $t_c > t_o$, each plaintext is expected to offer multiple solutions because $t_c - t_o$ extra bits have to be guessed before all of $P_j^\star$ is known. From the other side, similar observations can be made.

In Sect. 3.3, the parameters were $t_p = t_c = 16$ which results in $t_o = 16$. It is clear that the choice of these parameters influences both the time and memory complexity of the attack.

Complexity Analysis. The generalisation leads to a slightly more complex formula for expressing the time complexity of the attack. Because of the duality between guessing extra bits and filtering because of overlapping bits, all cases can be expressed in a single formula, which is a generalisation of (8) (i.e., with $t_p = t_c = 16$, it reduces to (8)):

$$2^{16}\left(32 \cdot 2^{16} + 2^{t_o}\left(2t_c \cdot 2^{16+t_c-t_o} + 2^{16+t_p-t_o}\left(2t_p + N_{\text{coll}} \cdot V\right)\right)\right) \quad . \tag{10}$$

In the generalised case, finding a collision is equivalent to finding an entry in a table of $16 + t_p - t_o$ elements that satisfies a t_o bit condition, so $N_{\text{coll}} = 2^{16+t_c-t_o}/2^{t_o}$. Verifying a collision now requires an average effort of

$$V = 2 \cdot \sum_{i=0}^{47-t_p-t_c} 2^{-i} + 2^{t_p+t_c-48} \cdot \left(528 + 528 \cdot 2^{-32}\right) \tag{11}$$

KeeLoq rounds. Simplification yields that the total complexity is equal to

$$32 \cdot 2^{32} + 2t_c \cdot 2^{32+t_c} + 2t_p \cdot 2^{32+t_p} + 4 \cdot 2^{80-t_p-t_c} + 528 \cdot 2^{32} \ . \qquad (12)$$

The optimum is found when $t_p = t_c = 15$ and thus $t_o = 14$, where the complexity reduces to $2^{53.524}$ KeeLoq rounds or $2^{44.5}$ full KeeLoq encryptions.

The memory requirements in the generalised case can also easily be evaluated. As before, 2^{16} plaintext/ciphertext pairs and 2^{16} values for X_i and Y_i are stored. The hash table now has $2^{16+t_p-t_o}$ entries of $64 + t_p$ bits each. For $t_p = t_c = 15$, the required memory is still less than 3 MB.

3.5 A Chosen Plaintext Attack

Using chosen plaintexts instead of known plaintexts, the attack can be improved. Consider the generalised attack from Sect. 3.4 in the case where $t_c < t_o$ (which is equivalent to $t_p > 16$). In this case, the $t_o - t_c$ least significant bits of the plaintext P_j are bits $(t_o, \ldots, t_c + 1)$ of $P_j^\star$. Hence, choosing the 2^{16} plaintexts in such a way that these $t_o - t_c$ least significant bits are equal to some constant, only 2^{t_c} guesses for $\underline{P_j^\star}$ have to be made at the beginning of the meet-in-the-middle step, instead of 2^{t_o}.

Complexity Analysis. As chosen plaintexts are only useful for the attack when $t_c < t_o$, we will only consider this case. The time complexity of the attack, in KeeLoq rounds, can be expressed as

$$2^{16}\left(32 \cdot 2^{16} + 2^{t_c}\left(2t_c \cdot 2^{16} + 2^{16+t_p-t_o}\left(2t_p + N_{\text{coll}} \cdot V\right)\right)\right) \ . \qquad (13)$$

The expected number of collisions is $N_{\text{coll}} = 2^{16}/2^{t_o}$. The verification cost, V, is given by (11). Simplification yields

$$32 \cdot 2^{32} + 2t_c \cdot 2^{32+t_c} + 2t_p \cdot 2^{48} + 4 \cdot 2^{80-t_p-t_c} + 528 \cdot 2^{32} \ . \qquad (14)$$

The optimum is found when $t_p = 20$, $t_c = 13$ and thus $t_o = 17$, where the attack has a time complexity of $2^{53.500}$ KeeLoq rounds or $2^{44.5}$ full KeeLoq encryptions. It is clear that the (theoretical) advantage over the known plaintext attack from Sect. 3.4 is not significant. However, as is discussed in the next section, the chosen plaintext variant can provide a significant gain in our practical implementation, because the verification cost V turns out to be higher there.

The memory complexity is about 2 MB as in Sect. 3.3 because the size of the hash table is the same. The data complexity remains at 2^{16} plaintext/ciphertext pairs, but note that we now require chosen plaintexts instead of known plaintexts.

4 Experimental Results

We have fully implemented and tested the attacks, using both simulated data and real data acquired from a HCS410 chip [14]. We made extensive use of bit slicing to do many encryptions in parallel throughout the implementation. However, because this parallelisation is not useful while verifying a collision, this verification step becomes more expensive in comparison. Hence, the optimal parameters for our implementation differ slightly from the theoretical ones. For the known plaintext attack from Sect. 3.4, the optimal parameters for our implementation were found to be $t_p = t_c = 16$. This means that, at least in our implementation, the best attack is the basic attack from Sect. 3.3. For the chosen plaintext attack, the optimal parameters are $t_p = 22$ and $t_c = 13$.

If we give the correct values for the 16 least significant key bits, the known plaintext attack completes in 10.97 minutes on average.[5] The chosen plaintext attack needs just 4.79 minutes to complete the same task.[6] This large difference can be explained by considering the impact of V, the cost of the verification step, on the time complexity of the attack. If V increases, and t_p and t_c are adapted as needed because their optimal values may change, the time complexity of the known plaintext attack increases much faster than the time complexity of the chosen plaintext attack does. Hence, even though their theoretical time complexities are the same, the chosen plaintext attack performs much better in our practical implementation because V is higher than the theoretical value.

We did not stop either of the attacks once a slid pair and the correct key were found, so we essentially tested the worst-case behaviour of the attack. This also explains the very small standard deviations of the measured running times. The machine used is an AMD Athlon 64 X2 4200+ with 1 GB of RAM (only one of the two CPU cores was used) running Linux 2.6.17. The attack was implemented in C and compiled with gcc version 4.1.2 (using the `-O3` optimiser flag). Critical parts of the code are written in assembly. Because the memory access pattern is random, but predictable to some extent, prefetching helped us to make maximum use of the cache memory.

The known plaintext attack performs over 288 times faster than the fastest attack with the same data complexity from [7,9], although the actual increase in speed is probably slightly smaller due to the difference in the machines used. Courtois et al. used (a single core of) a 1.66 GHz Intel Centrino Duo microprocessor [8]. The chosen plaintext attack performs more than 661 times faster, but this comparison is not very fair because chosen plaintexts are used. We note that the practicality of our results should also be compared with exhaustive key search due to the small key size. For the price of about 10 000 euro, one can obtain a COPACOBANA machine [12] with 120 FPGAs which is estimated to take about 1000 days to find a single 64-bit KeeLoq key.[7] Using our attack and

[5] We performed 500 experiments. The average running time was 658.15 s and the standard deviation was 1.69 s.

[6] We performed 500 experiments. The average running time was 287.17 s and the standard deviation was 0.55 s.

[7] The estimate was done by adapting the 17 days (worst case) required for finding a 56-bit DES key, taking into consideration the longer key size, the fact that more KeeLoq implementations fit on each FPGA, but in exchange take more clocks to test a key.

50 dual core computers (which can be obtained for roughly the same price), a KeeLoq key can be found in only two days.

5 Practical Applicability of the Attacks

5.1 Gathering Data

One might wonder if it is possible to gather 2^{16} known, or even chosen plaintexts from a practical KeeLoq authentication system. As mentioned in Sect. 2.2, a device like the HCS410 by Microchip Technology Inc. [14] supports two authentication protocols based on KeeLoq: "KeeLoq Hopping Codes" and "KeeLoq Identify Friend or Foe (IFF)". As the initial value of the counter used in "KeeLoq Hopping Codes" is not known, it is not easy to acquire known plaintexts from this protocol apart from trying all possible initial counter values. Also, since only 2^{16} plaintexts are ever used, knowing this sequence of 2^{16} ciphertexts suffices to break the system as this sequence is simply repeated.

The second protocol, "KeeLoq Identify Friend or Foe (IFF)" [14], is more appropriate for our attack. It is executed without any user interaction as soon as the transponder comes within the range of a decoder and is sent an activation signal. The challenges sent by the decoder are not authenticated in any way. Because of this, an adversary can build a rogue decoder which can be used to gather as many plaintext/ciphertext pairs as needed. The plaintexts can be fully chosen by the adversary, so acquiring chosen plaintexts is no more difficult than just known plaintexts. The only requirement is that the rogue decoder can be placed within the range of the victim's transponder for a certain amount of time. From the timings given in [14], we can conclude that one authentication completes within 60 ms or 90 ms, depending on the baud rate used. This translates into a required time of 65 or 98 minutes to gather the 2^{16} plaintext/ciphertext pairs. As these numbers are based on the maximum delay allowed by the specification [14], a real chip may respond faster, as our experiments confirm. No data is given with respect to the operational range in [14], because this depends on the circuit built around the HCS410 chip. However, one can expect the range to be short.

5.2 Key Derivation

The impact of the attack becomes even larger when considering the method used to establish the secret keys, as was previously noted by Bogdanov [6]. To simplify key management, the shared secret keys are derived from a 64-bit master secret (the manufacturer's code), a serial number and optionally a seed value [6,15,16]. The manufacturer's code is supposed to be constant for a large number of products (e.g., an entire series from a certain manufacturer) and the serial number of a transponder chip is public, i.e., it can easily be read out from the chip. The seed value is only used in the case of so-called "Secure Learning", and can also be obtained from a chip with relative ease [6,15,16]. The other option, "Normal Learning", does not use a seed value.

In both types of key derivation mechanisms, a 64-bit identifier is constructed, which contains the serial number, the (optional) seed and some fixed padding. Then, the secret key is derived from this identifier and the master secret using one of two possible methods. The first method simply uses XOR to combine the identifier and the master key. The consequence of this is that once a single key is known, together with the corresponding serial number and (optional) seed value, the master secret can be found very easily.

The second method is based on decryption with the KeeLoq block cipher. The identifier is split into two 32-bit halves which are decrypted using the KeeLoq block cipher, and concatenated again to form the 64-bit secret key. The master secret is used as the decryption key. Although much stronger than the first method, the master secret can still be found using a brute force search. Evidently, once the master secret is known, all keys that were derived from it are also compromised, and the security of the entire system falls to its knees. Thus, it is a much more interesting target than a single secret key. This may convince an adversary to legitimately obtain a car key, for the sole purpose of recovering the master key from its secret key.

6 Conclusion

In this paper we have presented a slide and meet-in-the middle attack on the KeeLoq block cipher which requires 2^{16} known plaintexts and has a time complexity of $2^{44.5}$ KeeLoq encryptions, and a variant using 2^{16} chosen plaintexts with the same theoretical time complexity.

We have fully implemented and tested both attacks. When given 16 key bits, the known plaintext attack completes successfully in 10.97 minutes. Due to implementation details, the chosen plaintext attack requires only 4.79 minutes when given 16 key bits. To the best of our knowledge, this is the fastest known attack on the KeeLoq block cipher.

Finally, we have shown that our attack can be used to attack real systems using KeeLoq due to the way it is intended to be used in practice. Moreover, one of the two suggested ways to derive individual Keeloq keys from a master secret is extremely weak, with potentially serious consequences for the overall security of systems built using the Keeloq algorithm.

Acknowledgements. We would like to thank Wim Aerts and Elke De Mulder for their help with the experiments. Also, we would like to thank the reviewers for their helpful comments.

References

1. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. Journal of Cryptology 7(4), 229–246 (1994)
2. Biryukov, A., Mukhopadhyay, S., Sarkar, P.: Improved Time-Memory Tradeoffs with Multiple Data. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 245–260. Springer, Heidelberg (2006)

3. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
4. Biryukov, A., Wagner, D.: Advanced Slide Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 586–606. Springer, Heidelberg (2000)
5. Bogdanov, A.: Cryptanalysis of the KeeLoq block cipher, Cryptology ePrint Archive, Report 2007/055, February 16 (2007), http://eprint.iacr.org/2007/055/
6. Bogdanov, A.: Attacks on the KeeLoq Block Cipher and Authentication Systems. In: 3rd Conference on RFID Security 2007 (RFIDSec 2007), http://rfidsec07.etsit.uma.es/slides/papers/paper-22.pdf
7. Courtois, N.T., Bard, G.V.: Algebraic and Slide Attacks on KeeLoq, Cryptology ePrint Archive, Report 2007/062, May 8 (2007), http://eprint.iacr.org/2007/062/
8. Courtois, N.T.: Personal communication (May 31, 2007)
9. Courtois, N.T., Bard, G.V., Wagner, D.: Algebraic and Slide Attacks on KeeLoq. In: Proceedings of Fast Software Encryption 2008, LNCS, Springer, Heidelberg (to appear)
10. Furuya, S.: Slide Attacks with a Known-Plaintext Cryptanalysis. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 214–225. Springer, Heidelberg (2002)
11. Hellman, M.E.: A Cryptanalytic Time-Memory Trade-Off. IEEE Transactions on Information Theory 26, 401–406 (1980)
12. Kumar, S., Paar, C., Pelzl, J., Pfeiffer, G., Schimmler, M.: Breaking Ciphers with COPACOBANA — A Cost-Optimized Parallel Code Breaker. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 101–118. Springer, Heidelberg (2006)
13. Microchip Technology Inc. KeeLoq® Authentication Products, http://www.microchip.com/keeloq/
14. Microchip Technology Inc., HCS410 KeeLoq® Code Hopping Encoder and Transponder Data Sheet, http://ww1.microchip.com/downloads/en/DeviceDoc/40158e.pdf
15. Microchip Technology Inc., AN642: Code Hopping Decoder using a PIC16C56, http://www.keeloq.boom.ru/decryption.pdf
16. Microchip Technology Inc., TB001: Secure Learning RKE Systems using KeeLoq Encoders, http://ww1.microchip.com/downloads/en/AppNotes/91000a.pdf
17. Wikipedia, KeeLoq (August 2007), http://en.wikipedia.org/wiki/KeeLoq

A Related-Key Attacks on KeeLoq

Related-key attacks [1] exploit the relations between the encryption processes under different but related keys.

In this appendix we present two related-key attacks on KeeLoq. The first attack is a very efficient attack using pairs of keys related by rotation. The second attack is an improvement of the attack presented in Sect. 3.3 using pairs of keys related by flipping the least significant bit of the key.

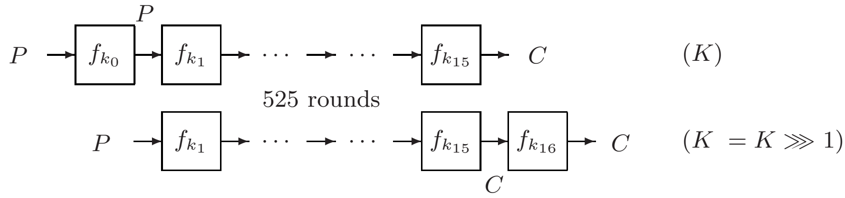


Fig. 5. A related-key attack using keys related by rotation

A.1 A Related-Key Attack Using Keys Related by Rotation

The first attack exploits the extremely simple way in which the key is mixed into the state during encryption.

Denote a full encryption of a plaintext P by KeeLoq with the key K by $E_K(P)$, and encryption through a single round with the subkey bit k by $f_k(P)$. Consider a pair (K, K') of related-keys, such that $K' = (K \ggg 1)$. If for a pair (P, P') of plaintexts we have $P' = f_{k_0}(P)$, where k_0 is the LSB of K, then $E_{K'}(P') = f_{k_{16}}(E_K(P))$. Indeed, in this case the encryption of P' under the key K' is equal to the encryption of P under K shifted by one round (see Fig. 5). This property, which is clearly easy to check, can be used to retrieve two bits of the secret key K.

Consider a plaintext P. We note that there are only two possible values of $f_{k_0}(P)$, i.e., $1||(P \gg 1)$ and $0||(P \gg 1)$. Hence, we ask for the encryption of P under the key K and for the encryption of the two plaintexts $P'_0 = 0||(P \gg 1)$ and $P'_1 = 1||(P \gg 1)$ under the related-key K', and check whether the ciphertexts satisfy the relation $E_{K'}(P') = f_{k_{16}}(E_K(P))$. This check is immediate, since $E_K(P)$ and $f_{k_{16}}(E_K(P))$ have 31 bits in common. Exactly one of the candidates (P'_0 or P'_1) is expected to satisfy the relation. This pair satisfies also the relation $P' = f_{k_0}(P)$.

At this stage, since P' and P are known, we can infer the value of k_0 immediately from the update rule of KeeLoq, using the relation $P' = f_{k_0}(P)$. Similarly, we can retrieve the value of k_{16} from the relation $E_{K'}(P') = f_{k_{16}}(E_K(P))$. Hence, using only three chosen plaintexts encrypted under two related-keys, we can retrieve two key bits with a negligible time complexity.

In order to retrieve additional key bits, we repeat the procedure described above with the pair of related-keys $(K', K'' = (K' \ggg 1))$ and one of the plaintexts P'_0 or P'_1 examined in the first stage. As a result, we require the encryption of two additional chosen plaintexts (under the key K''), and get two additional key bits: k'_0 and k'_{16}, which are equal to k_1 and k_{17}.

We can repeat this procedure 16 times to get bits $k_0, \ldots, k_{31}$ of the secret key. Then, the procedure can be repeated with the 16 related keys of the form $(K \ggg 32), (K \ggg 33), \ldots, (K \ggg 47)$ to retrieve the remaining 32 key bits. The attack then requires 66 plaintexts encrypted under 34 related keys (two plaintexts under each of 32 keys, and a single plaintext under the two remaining keys), and a negligible time complexity.

An option to reduce the required amount of plaintexts and related keys in exchange for a higher time complexity, is to switch to an exhaustive key search after a suitable number of key bits has been determined. For example, if 32 key bits remain to be found, a brute force search can be conducted in several hours on a PC, or even much less on FPGAs.

Another variant of the attack, requiring less related-keys, is the following. Denote the encryption of a plaintext P through r rounds of KeeLoq with the key $k = (k_0, \ldots, k_{r-1})$ by $f_k^r(P)$. Consider a pair of related-keys of the form $(K, K' = K \ggg r)$. If a pair of plaintexts (P, P') satisfies $P' = f_k^r(P)$, then the corresponding ciphertexts satisfy $E_{K'}(P') = f_{k'}^r(E_K(P))$, where $k' = (k_{16}, \ldots, k_{16+r-1})$. Since $E_K(P)$ and $f_{k'}^r(E_K(P))$ have $32 - r$ bits in common, this property is easy to check.

However, when $r > 1$, the task of detecting P' such that $P' = f_k^r(P)$ is not so easy. Actually, there are 2^r candidates for P', and hence during the attack we have to check 2^r candidate pairs. On the other hand, we can reduce the data complexity of this stage of the attack to $2^{1+r/2}$ by using structures: The first structure S_1 consists of $2^{r/2}$ plaintexts, such that the $32 - r$ least significant bits are equal to some constant C in all the plaintexts of the structure, and the other bits are arbitrary. The second structure S_2 also consists of $2^{r/2}$ plaintexts, such that the $32 - r$ most significant bits are equal to the same constant C in all the plaintexts of the structure, and the other bits are arbitrary. By birthday paradox arguments on the 2^r possible pairs (P, P') such that $P \in S_1$ and $P' \in S_2$ we expect one pair for which $P' = f_k^r(P)$, and this pair can be used for the attack.

In the attack, we go over the 2^r possible pairs and check whether the colliding bits of the relation $E_{K'}(P') = f_{k'}^r(E_K(P))$ are satisfied. If $r \leq 16$, this check discards immediately most of the wrong pairs. After finding the right pair, $2r$ bits of the key can be found using the algorithm presented in Sect. 3.2.

By choosing different values of r, we can get several variants of the attack:

1. Using $r = 16$, we can recover 32 key bits, and then the rest of the key can be recovered using exhaustive key search. The data complexity of the attack is 512 chosen plaintexts encrypted under two related-keys (256 plaintexts under each key), and the time complexity is 2^{32} KeeLoq encryptions.
2. Using $r = 8$ twice (for the pairs $(K, K \ggg 8)$, and $(K \ggg 8, K \ggg 16)$) we retrieve 32 key bits, and exhaustively search the remaining bits. The data complexity of the attack is 64 chosen plaintexts encrypted under three related-keys (16 plaintexts under two keys, and 32 plaintexts under the third key), and the time complexity is 2^{32} KeeLoq encryptions.
3. Using $r = 8$ four times (for the pairs $(K, K \ggg 8)$, $(K \ggg 8, K \ggg 16)$, $(K \ggg 32, K \ggg 40)$, and $(K \ggg 40, K \ggg 48)$) we can retrieve the full key. The data complexity of the attack is 128 chosen plaintexts encrypted under six related-keys (16 plaintexts under four keys, and 32 plaintexts under two keys), and the time complexity is negligible.

Other variants are also possible, and provide a trade-off between the number of chosen plaintexts and the number of related-keys.

A.2 Improved Slide/Meet-in-the-Middle Attack Using Related-Keys

Using a related-key approach, we can improve the attack presented in Sect. 3.3. Denote the encryption of a plaintext P through 64 rounds of KeeLoq under the key K by $g_K(P)$. Denote by e_0 the least significant bit of a word. We observe that if two related-keys (K, K') satisfy $K' = K \oplus e_0$, i.e., they differ in the least significant bit, and two plaintexts (P, P') satisfy $P' = P \oplus e_0$, then we have $g_K(P) = g_{K'}(P')$. Indeed, in the first round of encryption the key difference and the data difference cancel each other. As a result, after the first round the intermediate values in both encryptions are equal, and the key difference is not mixed into the data until the 65-th round. Thus, the intermediate values after 64 rounds are equal in both encryptions.

Now, recall that in Sect. 3.1, the pair (P_i, P_j) is called a slid pair if it satisfies $P_j = g_K(P_i)$. The attack searches among 2^{32} candidates for a slid pair, and then the key can be easily retrieved. Note that by the observation above, if (P_i, P_j) is a slid pair with respect to K, then the pair $(P_i \oplus e_0, P_j)$ is a slid pair with respect to $K' = K \oplus e_0$, and thus $E_{K'}(P_j) = g_{(K' \ggg 16)}(E_{K'}(P_i \oplus e_0))$. This additional slid pair can be used to improve the check of candidate slid pairs, and thus to reduce the time complexity of the attack.

More in detail, (10) can be rewritten as

$$2^{16}\left(48 \cdot 2^{16} + 2^{t_o}\left(3t_c \cdot 2^{16+t_c-t_o} + 2^{16+t_p-t_o}\left(3t_p + N_{\text{coll}} \cdot V\right)\right)\right) \quad . \tag{15}$$

The expected number of collisions becomes $N_{\text{coll}} = 2^{16+t_c-t_o}/2^{2t_o}$. Verifying a collision now costs on average V KeeLoq rounds, where

$$V = \sum_{i=0}^{47-t_p-t_c} \left(2 \cdot 2^{-2i} + 2^{-2i-1}\right) + 2^{2t_p+2t_c-96} \cdot \left(528 + 528 \cdot 2^{-32}\right) \quad . \tag{16}$$

Simplification yields:

$$48 \cdot 2^{32} + 3t_c \cdot 2^{32+t_c} + 3t_p \cdot 2^{32+t_p} + 3.33 \cdot 2^{96-2t_p-2t_c} + 528 \cdot 2^{32} \quad . \tag{17}$$

The optimum is situated at $t_p = t_c = 12$ where the time complexity of the attack is $2^{50.9}$ KeeLoq rounds, or $2^{41.9}$ full KeeLoq encryptions.

Summarising the attack, the data complexity is 2^{17} chosen plaintexts encrypted under two related-keys (2^{16} plaintexts under each key), and the time complexity is $2^{41.9}$ KeeLoq encryptions. The memory complexity is about 16 MB.

Key Recovery on Hidden Monomial Multivariate Schemes

Pierre-Alain Fouque[1], Gilles Macario-Rat[2], and Jacques Stern[1]

[1] École normale supérieure,
45 rue d'Ulm, 75005 Paris, France
{Pierre-Alain.Fouque, Jacques.Stern}@ens.fr
[2] Orange Labs
38–40, rue du Général Leclerc, 92794 Issy les Moulineaux Cedex 9, France
Gilles.Macariorat@orange-ftgroup.com

Abstract. In this paper, we study the key recovery problem for the C scheme and generalisations where the quadratic monomial of C (the product of two linearized monomials) is replaced by a product of three or more linearized monomials. This problem has been further generalized to any system of multivariate polynomials hidden by two invertible linear maps and named the Isomorphism of Polynomials (IP) problem by Patarin. Some cryptosystems have been built on this apparently hard problem such as an authentication protocol proposed by Patarin and a traitor tracing scheme proposed by Billet and Gilbert. Here we show that if the hidden multivariate system is the projection of a quadratic monomial on a base finite field, as in C , or a cubic (or higher) monomial as in the traitor tracing scheme, then it is possible to recover an equivalent secret key in polynomial time $O(n^d)$ where n is the number of variables and d is the degree of the public polynomials.

1 Introduction

Multivariate cryptography provides alternative schemes to RSA or DLog based cryptosystems where the underlying hard problem consists of solving a system of multivariate equations over a finite field. This problem is known to be NP-hard [13]. Moreover it seems to be interesting to build cryptosystems based on the assumption that it is hard, since contrary to the factorisation or the DLog problem, there is actually no known polynomial-time quantum algorithm to solve it, and generic algorithms that use Gröbner basis are exponential in time and memory. Finally, the proposed cryptosystems are very efficient in practice and can be implemented on low-cost smartcards since arithmetic on large integer is not required. Consequently, at the end of the nineties, a lot of multivariate cryptosystems were proposed.

One rich family of multivariate scheme is derived from a cryptosystem proposed by Matsumoto and Imai since 1988 and called C^*. Even though this scheme was broken by Patarin in 1995 [18], Patarin proposed various repairs. One of these repairs is the Minus transformation, suggested by Shamir in [23], which

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 19–30, 2008.

is a classical solution to avoid Patarin's or Gröbner basis attack. The SFLASH signature scheme, accepted by the NESSIE project in 2003, is a C^* scheme with this variation. Recently, SFLASH has been attacked by Dubois [illegible] in [7,6]. However, the attacks were not able to recover the secret key, as they rely on Patarin's attack which is [illegible] able to invert the public key.

The IP Problem. The corresponding key recovery problem, named the IP Problem, which stands for the Isomorphism of Polynomials has been introduced by Patarin since 1996 in [20] and studied later by Patarin, Goubin and Courtois in [22]. It can be stated as follows: given two sets of n polynomials in n variables A, B over a finite field $\mathbb{K}$ of q elements, find if there exist two linear and invertible mappings S and T in $\mathbb{K}$ such that $A = T \circ B \circ S$. This problem is not NP-hard provided the polynomial hierarchy does not collapse as proved by Faugère and Perret in [10]. However, if we relax S and T to be [illegible] linear mapping, then the problem is called MP, Morphism of Polynomials, and becomes NP-hard as shown in [22]. Finally, this problem is interesting since many Substitution Permutation Network block ciphers use as SBoxes a high degree monomial such as x^{254} in $GF(256)$ for the AES. Consequently, recovering the key for one round of the AES is equivalent to solve a special instance of the IP Problem, where the system B consists in 8 polynomials coming from a high degree monomial projected on $GF(2)$ and copied 16 times.

1.1 Related Work

Our method for solving the IP problem is not generic but is tailored to work for some cryptographic instances such as C^* based schemes or the traitor tracing scheme of [2]. For these cases, the algorithm is very efficient since it uses only linear algebra. The first step of our attack is similar to the recent attacks on SFLASH which can be extended to high degree monomials. In this case, we define high order differentials which have also been used in the cryptanalysis of symmetric schemes [16,15,14].

Previous Attacks on the IP problem. It is obvious that guessing S allows us to solve this problem since we can then compute the T function on some points and check whether it is a bijective linear mapping in time $O(n^3 q^{n^2})$.

If each polynomials of B only depends on a [illegible] number of variables such as 8 among the n in the case of the AES SBox, then polynomial time algorithms exist such as those described by Biryukov and Shamir in [4] or by Biham in [1].

However, when n is sufficiently large and each polynomial of B depends on many variables, the best known algorithm proposed so far by Patarin [illegible] has a complexity of $O(n^3 q^n)$. This last algorithm is very similar to the one proposed by Biryukov [illegible] in [3] in the context of linear equivalence problem for arbitrary permutations. In the case of SFLASH, where the set B is the projection of a quadratic monomial defined over $\mathbb{F}$ an extension of degree n of $\mathbb{K}$, then the Patarin [illegible] best algorithm has a complexity in $O(q^{n/2})$.

At Eurocrypt' 06, Faugère and Perret describe a Gröbner basis algorithm to solve the IP problem when B is a set of polynomials defined over a small number n of variables in an extension $\mathbb{F}$. Their algorithm is very efficient when the system of polynomials B is [illegible] and has small degree terms such as in the authentication scheme proposed by Patarin and some parameters of the traitor tracing scheme of Billet and Gilbert. However, for larger parameters proposed by Billet and Gilbert or for the parameters of SFLASH, the algorithm does not work. Their algorithm considers only terms of small degree in the system of polynomials so that the system they defined in the unknowns of S and T will be overdetermined. The complexity of this algorithm is dominated by the computation of a Gröbner basis for which we do not have complexity bound reflecting the practical behaviour. So, they conjecture that the complexity depends on the smallest value d so that there exists terms of degree d in B. For high degree monomial, as in the cases we consider in this paper, this parameter is exactly the degree of the monomials.

Differential Attack on SFLASH. As our attack relies on some information gained during the recent attacks on SFLASH, we informally describe here how they work.

Recently, some breakthrough results have been published on the cryptanalysis of the SFLASH signature scheme by Dubois [illegible] in [7,6]. SFLASH comes from the C^* family, [illegible] the internal quadratic monomial of the form $P(x) = x^{1+q^\theta}$ over an extension $\mathbb{F}$ of degree n of the base finite field $\mathbb{K}$ is hidden by two linear bijective mappings S and T. The public key is $\mathbf{P} = T \circ P \circ S$ and if some polynomials of the public key are removed, we get a SFLASH public key. In [7], the authors consider the case where $\gcd(\theta, n) > 1$.

The basic idea of [14,7,6] is to recover some of these polynomials or equivalent polynomials by noticing that the internal polynomial $P \circ S$ over $\mathbb{F}$ forms a set of n polynomials over $\mathbb{K}$. Then, the action of T consists of linear combination of these n polynomials. Consequently, if we are able to recover other linear combinations of these polynomials with independent coefficients, we will be able to recover a complete public key.

The last results show that it is possible to reconstruct equivalent missing polynomials using only 3 polynomials of the public key. The way to do it is to reconstruct some special linear applications related to the secret S, of the form $N_u = S^{-1} M_u S$ so that M_u denotes the multiplications by u in $\mathbb{F}$. In [7], it is shown that the maps N_u where u are solutions of $x^{q^\theta} + x = 0$ are easy to recover using a linear characterization, whereas in [6], more involved analysis are needed. However, this last attack is more powerful since any multiplication can be recovered. Then, the composition of these maps N_u with the public key $\mathbf{P}$ is of the form $T \circ P \circ M_u \circ S$ and since P is multiplicative, $\mathbf{P} \circ N_u$ is of the form $T' \circ P \circ S$ and if T' contains rows independent of those of T, then we get new polynomials of the public key which will be independent from the first ones. Finally, once the public key is recovered, Patarin's attack can be applied.

Consequently, in this paper we can assume that no equation is removed.

1.2 Our Results

In this paper, we show that the recent attacks on multivariate schemes can be made more devastating and lead to total break of the C^* schemes family. More precisely, we show that the IP problem for C^* is easy and we can recover secret keys S and T or equivalent can be recovered given a $N_u = S^{-1}M_uS$ linear mappings. Indeed, these matrices depend on the secret S, but M_u are unknown. Here, we show how we can recover u and then, how we can recover S' and T'. This last step is not always easy and when $\gcd(n, \theta) > 1$, many parasitic solutions can exist. For the SFLASH signature scheme, the recent attacks rely on Patarin's attack in their final stage. However, this attack can become exponential in some bad cases. Here, our attack on the C^* schemes family is always polynomial to recover the secret key and can be seen as a new attack on the C^* scheme.

Moreover, we show that for high degree monomials, we can also recover the matrices N_u as in the case of the quadratic monomials of SFLASH and recover the secret keys. To get a linear characterization of N_u, we use high order differentials as an analog to symmetric cryptanalysis. These two results improve on a result of Faugère and Perret at Eurocrypt '06 using Gröbner basis [10] which solves only some particular cases but not all the proposed parameters by Billet and Gilbert. For the C^* case, Faugère and Perret indicate that their approach cannot take into account SFLASH parameters since the system of polynomials is too sparse. Here, we only present polynomial time attack to recover these values for SFLASH and the second parameter proposed by Billet and Gilbert in the case of the traitor tracing scheme [2].

1.3 Organization of the Paper

In section 2 we present the problem Isomorphism of Polynomials which represents the key recovery problem in multivariate schemes. Then, we present the differential of the public key which allows to give a characterization of the interesting linear mappings we are looking for. Then, we show how to solve the IP problem when the internal polynomial is a monomial in section 4. In section 5, we show that the SFLASH public key can be recovered in all cases and on monomial of higher degree of the traitor tracing scheme before the conclusion.

2 Isomorphisms of Polynomials Problem (IP)

In this section, we present the Isomorphism of Polynomials problem stated by Patarin in [20,22]. It has been used by Billet and Gilbert in [2] to define a traitor tracing scheme.

2.1 Description of the IP Problem

The IP Problem is defined for any two sets A, B of n multivariate polynomials and the problem is to find S and T two linear and bijective maps on n variables so that $A = T \circ B \circ S$. In this paper, we focus on special instances of this problem when the system B is the projection on the base field $\mathbb{K}$ of a polynomial defined over an extension of degree n of $\mathbb{K}$.

Let $\mathbb{K}$ be a small finite field of q elements and $\mathbb{F}$ an extension of degree n over $\mathbb{K}$. Let π be an isomorphism from $\mathbb{K}^n$ onto $\mathbb{F}$ and P some polynomial over $\mathbb{F}$. Then, let S and T be two linear or affine invertible transformations over $\mathbb{K}^n$. The maps S and T are kept secret. Finally let $\mathbf{P} = T \circ \pi^{-1} \circ P \circ \pi \circ S$ be a set of n polynomial forms over $\mathbb{K}^n$. This system of multivariate polynomials $\mathbf{P}$ is also named the public key. The problem can now be expressed as follows:

IP Problem. $\mathbb{K}$ n P $\mathbf{P}$ S' T' $\mathbb{K}^n$ π' $\mathbb{K}^n$ n $\mathbb{K}$

$$\mathbf{P} = T' \circ \pi'^{-1} \circ P \circ \pi' \circ S'.$$

The choice of π' is indifferent. Indeed, should we choose $\tilde{\pi}$, then there exists some change of coordinates such that $\varphi = \tilde{\pi}^{-1} \circ \pi'$. If $(T',\, S',\, \pi')$ is a solution, then $(\tilde{T} = T' \circ \varphi^{-1},\, \tilde{S} = \varphi \circ S',\, \tilde{\pi})$ is another solution.

In the sequel, by some misuse of language, we avoid writing the isomorphism π and its inverse π^{-1} when their use is obvious and simply write $\mathbf{P} = T \circ P \circ S$.

IP with Polynomials. In this article, we mainly study the case where P is a monomial of the form $P(x) = x^{1+q^{\theta_1}+\ldots+q^{\theta_{d-1}}}$ defined over an extension field $\mathbb{F}$ of degree n of $\mathbb{K}$. If we project this monomial over the base field $\mathbb{K}$, we get n multivariate polynomial of degree d since the mappings $x \mapsto x^{q^i}$ for integers i are $\mathbb{K}$-linear. Consequently, the changes between the public key $\mathbf{P}$ and the internal polynomial P are changes of variables, which do not modify the degree of the multivariate polynomials.

2.2 Equivalent Keys

Solutions to the IP Problem are in fact not unique. See [24] for a discussion about equivalent keys. For instance, let's analyze the case $P(x) = x^{1+q^\theta}$. Let's note M_u (multiplications) and φ_i (Frobenius) defined by $M_u(x) = ux$ and $\varphi_i(x) = x^{q^i}$. So if (T', S') is a solution then so are

$$(T' \circ \pi^{-1} \circ M_{1/u^{q^\theta+1}} \circ \pi, \pi^{-1} \circ M_u \circ \pi \circ S') \text{ and } (T' \circ \pi^{-1} \circ (\varphi_i)^{-1} \circ \pi, \pi^{-1} \circ \varphi_i \circ \pi \circ S').$$

3 Differential and Properties for Monomials

The differential of the public key of a multivariate scheme has been introduced in a systematic cryptanalytic method by Fouque in [11]. Later, this method has been developed and extended in [8,9,7,6] to attack various systems.

3.1 Differential of Polynomials

For a general polynomial P, the differential in some point a, denoted by D_aP, is formally defined by:

$$D_aP(x) = P(x+a) - P(x) - P(a) + P(0).$$

We may also refer it as $DP(x,a)$ which is symmetric since $D_aP(x) = D_xP(a)$. The later notation also represents the fact that the differential is a bilinear expression and consequently, it can be represented by a matrix. In our case, all polynomials of the public key can be represented as a bilinear mapping.

The interest of studying the differential is that it "lowers" the degree and it is homogeneous. For instance, if $\deg(\mathbf{P}) = 2$ then $\deg(D_a\mathbf{P}) = 1$ and $D_a\mathbf{P}$ is linear. In this case, the differential acts as it "kills" the parts of degree 1 and 0 of $\mathbf{P}$.

Differential of Monomials of Higher Degree. For higher degrees, we may define differentials of higher order. For instance, if $\deg(\mathbf{P}) = 3$: $D_{a,b}P(x) = D_a(D_bP(x))$ defines a second order differential and $\deg(D_{a,b}\mathbf{P}(x)) = 1$. We may also note it $DP(a,b,x)$ for the same reason as previously.

Differential of the Public Key. Let us study how the differential operates on the public key. We assume here that $P(x) = x^{1+q^\theta}$. First, if S and T are linear, then we have

$$D_a\mathbf{P}(x) = T(D_{S(a)}P(S(x))) \tag{1}$$

Taking into Account the Affine Parts. If S and T are affine, we denote by Σ_c the addition with c. With this notation, we have: $(P \circ \Sigma_c)(x) = P(x) + xc^{q^\theta} + x^{q^\theta}c + P(c)$. Now, we can easily express that $D_a(P \circ \Sigma_c)(x) = D_aP(x)$, since $xc^{q^\theta} + x^{q^\theta}c + P(c)$ is affine. Since $S(x) = DS(x) + S(0)$ and $P \circ S = P \circ \Sigma_{S(0)} \circ DS$, we deduce a similar relation: $D_a\mathbf{P}(x) = DT(D_{DS(a)}P(DS(x)))$. So, the previous relation is just like relation (1) where S and T are replaced by their linear part DS and DT.

3.2 Multiplicative Property of the Differential

In this section, we show that a characterization equation exists for hidden monomials that involves a linear mapping N. Since the equation is linear in the unknown of N and depends only on the public key, N can be easily found.

Multiplicative Property for SFLASH. For $P(x) = x^{1+q^\theta}$ there is an interesting property of the differential:

$$D_xP(M_u(y)) + D_yP(M_u(x)) = M_{u+u^{q^\theta}}(D_yP(x)) \tag{2}$$

where M_u is the multiplication by u in $\mathbb{F}$. We can also rewrite this equation as $DP(xu,y) + DP(x,yu) = (u + u^{q^\theta})DP(x,y)$. How is this property (2) transfered to the public system? Firstly for the sake of simplicity, we may assume that S and T are linear. Otherwise, we will see that considering only their linear part is a good approach when they are affine.

If we denote by N_u the conjugate by S of M_u, namely $N_u = S^{-1} \circ M_u \circ S$, property (2) becomes:

$$\begin{aligned} D_x\mathbf{P}(N_u(y)) + D_y\mathbf{P}(N_u(x)) &= T(M_{u+u^{q^\theta}}(D_{S(y)}F(S(x)))) \\ &= (T \circ M_{u+u^{q^\theta}} \circ T^{-1})(D_y\mathbf{P}(x)) \end{aligned} \tag{3}$$

If we consider the vector space of symmetric bilinear forms such that $b(x,x) = 0$ of dimension $n(n-1)/2$, then the bilinear forms of the left hand side are in the vector space V spanned by the bilinear forms of the differential of the public key $D_y\mathbf{P}(x)$ of dimension n. This equation is linear in the n^2 unknowns of N_u and stating that one quadratic form of the LHS is in this vector space gives $n(n-1)/2$ linear equations and n additional unknowns. Therefore, expressing that 3 forms of the LHS are in V is sufficient to completely determine N_u.

Multiplicative Property for Higher Degree. For degree 3 or 4, similar expressions for this property can be derived, by considering respectively:

$$D_{x,y}\mathbf{P}(N_u(z)) + D_{x,z}\mathbf{P}(N_u(y)) + D_{y,z}\mathbf{P}(N_u(x)), \tag{4}$$

$$D_{x,y,z}\mathbf{P}(N_u(v)) + D_{x,y,v}\mathbf{P}(N_u(z)) + D_{x,z,v}\mathbf{P}(N_u(y)) + D_{y,z,v}\mathbf{P}(N_u(x))). \tag{5}$$

In case (4), we get trilinear forms and the multiplication by $u + u^{q^\theta}$ is replaced by $u + u^{q^{\theta_1}} + u^{q^{\theta_2}}$ for degree 3 and by $u + u^{q^{\theta_1}} + u^{q^{\theta_2}} + u^{q^{\theta_3}}$ for degree 4.

Multiplicative Property is a Characterization. The property (2) and the ones infered for higher degree are a characterization. Indeed the only linear mappings M and M' satisfying:

$$D_xP(M(y)) + D_yP(M(x)) = M'(D_yP(x)) \tag{6}$$

are the multiplications.

The idea of the proof is that the $\mathbb{K}$-linear applications over $\mathbb{F}$ can be expressed as linearized polynomials such as $M(x) = \sum_{i=0}^{n-1} \lambda_i x^{q^i}$ where coefficients λ_i belong to $\mathbb{F}$. By replacing this expression in equation (6), provided that n is large enough, all coefficients λ_i must be null except λ_0. Hence the result $M(x) = \lambda_0 x$.

This result is true only if n is not too close to d. When n is too small, there is a side effect that allows linear applications other than multiplications to be solution of equation (6). Experimentally, we have found the lower limit of n according to d. For $d = 2$ and $d = 3$, we must have $n \geq 5$. For $d = 4$, we must have $n \geq 7$.

4 Recovering S and T

The basic idea to recover equivalents for S and T is to find some N_u and use equation: $N_u = S^{-1}M_uS$. If we can recover u, then M_u is known and we can linearized it to $SN_u = M_uS$, where S is the unknown we are looking for.

Description of the Attack. In the following, we describe the different steps of the attack to recover equivalent S and T.

1. Find all linear transformations L such as $D_x\mathbf{P}(L(y)) + D_y\mathbf{P}(L(x))$ is a set of bilinear forms, all of them being linear combinations of the elements of $D_y\mathbf{P}(x)$. Due to the characterization, the space of solutions is the conjugate by S of the multiplications.
2. Pick up at random one solution L which characteristic polynomial is irreducible over $\mathbb{K}$.
3. Find u such as L and M_u are conjugate. Since L and M_u must have the same characteristic polynomial, choose u as any root of the characteristic polynomial of L. Since characteristic polynomial is irreducible over K, roots are primitive elements of $\mathbb{F}$.
4. Solve the linear system $X.L = M_u.X$ where the unknown X is a linear mapping of $\mathbb{K}^n$.
5. Pick up at random any non trivial solution S.
6. Compute T as $\mathbf{P} \circ S^{-1} \circ P^{-1}$.

Recovering L. In [7,6], it is described how the first step of this attack can be mounted since systems in step 1 is overdefined. Consequently, only a few coordinates of $D_y\mathbf{P}(x)$ are sufficient to solve it. This is the same reason why the "Minus" scheme of SFLASH can be defeated even if some public polynomial are removed.

It is also possible to reconstruct S and T even though they are affine. The computations are the same, but we replace $\mathbf{P}$ by $D\mathbf{P}$. At steps 5 and 6, we can find actually the linear parts of S and T, that is DS and DT. Then, using equation:

$$(DT)^{-1} \circ D\mathbf{P}(x) = D(F \circ S)(x) = (DS(x))^{1+q^\theta} + (DS(x))^{q^\theta} S(0) + DS(x)S(0)^{q^\theta}$$

replace x by random values, in order to gain enough linear independent equations, all of the form $ay^{q^\theta} + by + c = 0$, and find the solution $S(0)$. Then, compute $T(0) = \mathbf{P}(0) - (DT \circ P \circ S)(0)$.

Recovering M_u. To recover M_u, we first show how we recover u. Since L is the conjugate of M_u by the secret matrix S, they are similar and so, they have the same minimal polynomial. Furthermore, u is a root of the minimal polynomial of M_u [1]. Indeed, if Π is the minimal polynomial of M_u, then $\Pi(M_u) = 0$ and so $\Pi(ux) = 0$ for all x, and so $\Pi(u) = 0$ for $x = 1$. Moreover, it is also well-known

[1] In fact, one can prove that u and M_u have the same minimal polynomial.

that the roots of a minimal polynomial are conjugates, [illegible] are the elements $\{u, u^q, u^{q^2}, \ldots, u^{q^{n-1}}\}$. This result can be easily seen since the coefficients of the minimal polynomial belong to $\mathbb{F}_q$, and for any element α of $\mathbb{F}_q$, we have $\alpha^{q^i} = \alpha$, thus for the minimal polynomial p of u, $p(u^{q^i}) = p(u)^{q^i} = 0$. The conjugate property stands also for matrices, since $M_u = (\varphi_q^i)^{-1} M_{u^{q^i}} \varphi_q^i$, where $\varphi_q^i(x) = x^{q^i}$ is the ith frobenius map. Therefore, even though we do not choose the right conjugate, since the frobenius application commutes with the internal monomial, we will always find equivalent secret keys. So, once L is known, it suffices to select any of the roots of its minimal polynomial as value for u.

Equivalent Keys and Space of Solutions. At step 1, solutions should be a subspace of dimension n, isomorphic to $\mathbb{F}$, since it is the conjugate by S of the space of multiplication matrices. For instance, trivial solutions are diagonal matrices which correspond to elements of $\mathbb{K}$. So at this step we just need to select any matrix corresponding to a multiplication by a primitive element of $\mathbb{F}$. At step 3, roots of the characteristic polynomial are conjugate, since it is irreducible over $\mathbb{K}$ and its coefficients belong to $\mathbb{K}$. Thus selecting u^{q^i} instead of u is equivalent to multiply the solutions by φ_i. At step 5, solutions can be obtained from a particular one, by multiplying it by any multiplication matrix M_u.

[illegible] In the wording of the IP problem, we can assume that P is unknown, only its degree is known, since the number of monomials of a given degree is small.

5 Applications

The following experimental results have been obtained with an Opteron 850 2.2GHz, with 32 GBytes of Ram. The systems associated with the instance of the problems and their solutions have been generated using the Magma software, version 2.13-15.

If the following tables, t_{gen} is the time for computing the coefficient of the problem, mainly the linear application that gives $D_x\mathbf{P}(L(y)) + D_y\mathbf{P}(L(x))$ for any L, at step 1, t_{sol} is the time for solving the problem, which is basically a linear algebra issue, regarding intersection of subspaces. 's.' and 'm.' denote respectively second and minute.

5.1 SFLASH Signature Scheme

The following results concern a general instance of the IP problem for an homogeneous C*scheme of degree 2, that is we are looking for linear S and T. Nevertheless, this is almost the problem of key recovery for the SFLASH Signature scheme, where some coordinates (equations) are missing, since finding M_u enables to regenerate missing coordinates.

q	d	n	t_{gen}	t_{sol}
2^{16}	2	19	0.4 s.	0.5 s.
2^{16}	2	21	0.6 s.	1 s.
2^7	2	37	6 s.	23 s.
2	2	67	55 s.	10 s.
2^7	2	67	60 s.	12 m.

The first row corresponds to the second challenge of Billet and Gilbert. Faugère and Perret in [10] were unable to solve it and conjectured that the system was too sparse. Moreover, row 3 and 5 correspond to the practical instances of SFLASH v2 and SFLASH v3. In this case, the number of variables is too large and Gröbner basis algorithm cannot take into account such parameters. However, contrary to [10], our approach can only deal with internal system of multivariate scheme coming from the projection of a monomial and not any polynomials. In the case of SFLASH parameters, we do not give the value r of the removed equations since previous attacks [7,6] can always be used to recover missing polynomials of the public key.

5.2 Traitor Tracing of Billet and Gilbert

Here as above, the results concern a general instance of the IP problem for an homogeneous C* scheme, but of degree 3 and 4. The change was in the use of the expressions (4), and (5).

q	d	n	t_{gen}	t_{sol}	θ_1	θ_2	θ_3
2^9	3	10	0.6 s.	0.1 s.	1	4	
2^9	3	18	12 s.	5 s.	1	6	
2^9	3	19	15 s.	7 s.	1	4	
2^9	3	20	20 s.	11 s.	1	4	
2^9	3	21	26 s.	15 s.	1	6	
2^{16}	4	7	0.2 s.	0.2 s.	1	2	6
2^{16}	4	8	0.65 s.	0.4 s.	1	3	7
2^{16}	4	9	1.4 s.	0.3 s.	1	2	7
2^8	4	10	11 s.	8 s.	1	3	5
2^8	4	11	19 s.	44 s.	1	2	6
2^8	4	12	32 s.	80 s.	1	2	10

In these experiments, we give the values of θ_1, θ_2 and θ_3 such that the monomials can be inverted and so that there is no intermediate finite field of $\mathbb{F}$, $\gcd(\theta_1, \theta_2, n) = 1$. We can remark that from $n = 7$ for $d = 4$, we can solve the IP problem for monomials more efficiently than [10]. These results confirm experimentally the complexity of the resolution of the problem, namely $O(\log(q)^2 n^d)$. We can finally remark that the degree d is exactly the heuristic value given by Faugère and Perret in the case of high degree monomials defined over an extension field.

5.3 The ℓ-IC Scheme

At PKC'07, Ding et al. presented a new multivariate scheme based on Cremona maps in [5]. This scheme has been attacked at PKC'08 by Fouque et al. in [12]. In this attack, the authors are also able to recover equivalent secret keys. The way they recover u consists in raising N_u to some power so that u^α has a small order and then, exhaustive search can be performed. Fortunately, for the proposed parameters, it is always the case. However, if this trick is not possible, our method that computes the minimal polynomial can be done and we get directly the value u. Consequently, we can improve the cryptanalysis of the ℓ-IC scheme.

6 Conclusion

Here, we describe a key recovery attack on the C*schemes family which lead to the recovery of equivalent secret keys. This means that an attacker would be in the same position than a legitimate user. Moreover, this attack is polynomial in time and space, and so it is very practical and can be executed within few seconds on the recommended values of the parameters of the schemes.

Acknowledgements

The first and last authors are very grateful to Adi Shamir and Vivien Dubois for interesting discussions on this subject.

The work described in this paper has been supported by the ANR MAC project and by the European Commission through the IST Program under contract IST-2002-507932 ECRYPT.

References

1. Biham, E.: Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R). In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 408–416. Springer, Heidelberg (2000)
2. Billet, O., Gilbert, H.: A Traceable Block Cipher. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 331–346. Springer, Heidelberg (2003)
3. Biryukov, A., De Canniére, C., Braeken, A., Preneel, B.: A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 33–50. Springer, Heidelberg (2003)
4. Biryukov, A., Shamir, A.: Structural Cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 394–405. Springer, Heidelberg (2001)
5. Ding, J., Wolf, C., Yang, B.-Y.: ℓ-Invertible Cycles for Multivariate Quadratic Public Key Cryptography. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 266–281. Springer, Heidelberg (2007)
6. Dubois, V., Fouque, P.-A., Shamir, A., Stern, J.: Practical Cryptanalysis of SFLASH. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, Springer, Heidelberg (2007)

7. Dubois, V., Fouque, P.-A., Stern, J.: Cryptanalysis of SFLASH with Slightly Modified Parameters. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 264–275. Springer, Heidelberg (2007)
8. Dubois, V., Granboulan, L., Stern, J.: An Efficient Provable Distinguisher for HFE. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 156–167. Springer, Heidelberg (2006)
9. Dubois, V., Granboulan, L., Stern, J.: Cryptanalysis of HFE with Internal Perturbation. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 249–265. Springer, Heidelberg (2007)
10. Faugére, J.-C., Perret, L.: Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 30–47. Springer, Heidelberg (2006)
11. Fouque, P.A., Granboulan, L., Stern, J.: Differential Cryptanalysis for Multivariate Schemes. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 341–353. Springer, Heidelberg (2005)
12. Fouque, P.A., Macario-Rat, G., Perret, L., Stern, J.: Total Break of the ℓ-IC Signature Scheme. In: PKC 2008. LNCS, vol. 4939, pp. 1–17. Springer, Heidelberg (2008)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of *NP*-Completeness. Freeman, New-York (1979)
14. Gilbert, H., Minier, M.: Cryptanalysis of SFLASH. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 288–298. Springer, Heidelberg (2002)
15. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1994)
16. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Proc. Symposium on Communication, Coding and Cryptography (1994)
17. Matsumoto, T., Imai, H.: Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
18. Patarin, J.: Cryptoanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt 1988. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
19. Patarin, J.: Asymmetric Cryptography with a Hidden Monomial. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 45–60. Springer, Heidelberg (1996)
20. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
21. Patarin, J., Courtois, N., Goubin, L.: FLASH, a Fast Multivariate Signature Algorithm. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 298–307. Springer, Heidelberg (2001)
22. Patarin, J., Goubin, L., Courtois, N.: Improved Algorithms for Isomorphisms of Polynomials. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 184–200. Springer, Heidelberg (1998)
23. Shamir, A.: Efficient Signature Schemes Based on Birational Permutations. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 1–12. Springer, Heidelberg (1994)
24. Wolf, C., Preneel, B.: Equivalent Keys in HFE, C , and Variations. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 33–49. Springer, Heidelberg (2005)

Predicting Lattice Reduction

Nicolas Gama and Phong Q. Nguyen

École normale supérieure/CNRS/INRIA, 45 rue d'Ulm, 75005 Paris, France
nicolas.gama@ens.fr
http://www.di.ens.fr/~pnguyen

Abstract. Despite their popularity, lattice reduction algorithms remain mysterious cryptanalytical tools. Though it has been widely reported that they behave better than their proved worst-case theoretical bounds, no precise assessment has ever been given. Such an assessment would be very helpful to predict the behaviour of lattice-based attacks, as well as to select keysizes for lattice-based cryptosystems. The goal of this paper is to provide such an assessment, based on extensive experiments performed with the NTL library. The experiments suggest several conjectures on the worst case and the actual behaviour of lattice reduction algorithms. We believe the assessment might also help to design new reduction algorithms overcoming the limitations of current algorithms.

Keywords: Lattice Reduction, BKZ, LLL, DEEP Insertions, Lattice-based cryptosystems.

1 Introduction

Lattices are discrete subgroups of $\mathbb{R}^n$. A lattice L can be represented by a basis, that is, a set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ in $\mathbb{R}^n$ such that L is equal to the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_d) = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z} \right\}$ of all integer linear combinations of the $\mathbf{b}_i$'s. The integer d is the dimension of the lattice L. A lattice has infinitely many bases, but some are more useful than others. The goal of [illegible] is to find interesting lattice bases, such as bases consisting of reasonably short and almost orthogonal vectors.

Lattice reduction is one of the few potentially hard problems currently in use in public-key cryptography (see [29,23] for surveys on lattice-based cryptosystems), with the unique property that some lattice-based cryptosystems [3,34,35,33,11] are based on worst-case assumptions. And the problem is well-known for its major applications in public-key cryptanalysis (see [29]): knapsack cryptosystems [32], RSA in special settings [7,5], DSA signatures in special settings [16,26], [illegible] One peculiarity is the existence of very efficient approximation algorithms, which can sometimes solve the exact problem. In practice, the most popular lattice reduction algorithms are: floating-point versions [37,27] of the LLL algorithm [20], the LLL algorithm with deep insertions [37], and the BKZ algorithms [37,38], which are all implemented in the NTL library [39].

Although these algorithms are widely used, their performances remain mysterious in many ways: it is folklore that there is a gap between the theoretical

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 31–51, 2008.

analyses and the experimental performances. In the eighties, the early success of lattice reduction algorithms in cryptanalysis led to the belief that the strongest lattice reduction algorithms behaved as perfect oracles, at least in small dimension. But this belief has shown its limits: many NP-hardness results for lattice problems have appeared in the past ten years (see [23]), and lattice-based attacks rarely work in very high dimension. Ten years after the introduction of the NTRU cryptosystem [15], none of the NTRU challenges has been solved, the smallest one involving a lattice of dimension 334. On the other hand, all five GGH-challenges [12] have been solved [25], except the 400-dimensional one. It is striking to see that the GGH-350 challenge has been solved, while no 334-dimensional NTRU lattice has ever been solved. The behaviour of lattice algorithms is much less understood than that of their factoring and discrete logarithm counterpart. It would be useful to have at least a model (consistent with experiments) for the performances of existing lattice algorithms.

Our Results. We provide a concrete picture of what is achievable today with the best lattice reduction algorithms known in terms of output quality and running time, based on extensive experiments performed with the NTL library during the past year. This sheds new lights on the practical hardness of the main lattice problems, and allows to compare the various computational assumptions (Unique-SVP, Approximate-SVP) used in theoretical lattice-based cryptography [33,11,35,34,3]. For instance, our experiments strongly suggest that Unique-SVP is significantly easier than Approximate-SVP, and that the hardness of Approximate-SVP depends a lot on the structure of the lattice. Our experiments also clarify the gap between the theoretical analyses and the experimental performances of lattice algorithms, and point out several surprising phenomenons on their behaviour. The most important fact is that asymptotically, all the algorithms known seem to only achieve an exponential approximation factor as predicted by theory, but the exponentiation bases turn out to be extremely close to 1, much closer than what theory is able to prove. This seems to nullify the security property of cryptosystems based on the hardness of approximating lattice problems with big polynomial factors, unless such schemes use large parameters. On the other hand, it also makes clear what are the limits of today's algorithms: in very high dimension, today's best algorithms roughly square root the exponential approximation factors of LLL. Our predictions may explain in retrospect why the 350-dimensional GGH lattice has been solved, but not the 334-dimensional NTRU lattices or the 400-dimensional GGH lattice. We believe the assessment might help to design new reduction algorithms overcoming the limitations of current algorithms. As an illustration, we present an alternative attack on the historical NTRU-107 lattices of dimension 214.

Related work. The NTRU company has performed many experiments with BKZ to evaluate the cost of breaking NTRU lattices. However, such experiments only dealt with NTRU lattice bases, which have a very special structure. And their experiments do not lead to any prediction on what can be achieved in general. Our work is in the continuation of that of Nguyen and Stehlé [28] on the average-case of LLL. But the goal of this paper is to provide a much broader picture: [28] only performed experiments with LLL (and not improved algorithms like BKZ which are much more expensive), and focused on the so-called Hermite-SVP problem, without considering cryptographic lattices with special structure.

Road map. The paper is organized as follows. In Section 2, we provide necessary background on lattice reduction. In Section 3, we provide a concrete picture of what lattice reduction algorithms can achieve today. In Section 4, we analyze the experimental running time of lattice reduction algorithms, and point out several unexpected phenomenons. In Section 5, we compare our predictions with former experiments on GGH and NTRU lattices.

2 Background

We refer to [29,23] for a bibliography on lattices.

2.1 Lattices

In this paper, by the term lattice, we mean a discrete subgroup of some $\mathbb{R}^m$. Lattices are all of the form $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^n m_i \mathbf{b}_i \mid m_i \in \mathbb{Z}\}$ where the $\mathbf{b}_i$'s are linearly independent vectors. Such n-tuple of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is called a [illegible] of the lattice: a basis will be represented by a row matrix. The [illegible] of a lattice L is the dimension n of the linear span of L. The [illegible] of k vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is $\det\left(\langle \mathbf{v}_i, \mathbf{v}_j \rangle\right)^{1/2}_{1 \le i,j \le k}$. The [illegible] $\mathrm{vol}(L)$ (or [illegible]) of a lattice L is the volume of any basis of L.

Minima. We denote by $\lambda_i(L)$ the i-th minimum of a lattice L: it is the radius of the smallest zero-centered ball containing at least i linearly independent lattice vectors. The so-called Hermite's constant γ_n of dimension n satisfies Minkowski's second theorem: for any n-dimensional lattice L, and for any $1 \le d \le n$, we have

$$\left(\prod_{i=1}^{d} \lambda_i(L)\right)^{1/d} \le \sqrt{\gamma_n}\,\mathrm{vol}(L)^{1/n}.$$

The exact value of γ_n is only known for $1 \le n \le 8$ and $n = 24$. For other values of n, the best numerical upper bounds known are given in [6]. Asymptotically, Hermite's constant grows linearly in n. Rankin (see [8]) generalized the minima $\lambda_i(L)$ to the smallest subvolumes: $\gamma_{n,m}(L)$ is the minimal value of $\mathrm{vol}(\mathbf{x}_1, \ldots, \mathbf{x}_m)/\mathrm{vol}(L)^{m/n}$ where $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ range over all m linearly independent lattice vectors.

Random lattices. There is a beautiful albeit mathematically sophisticated notion of random lattice, which follows from Haar measures of classical groups. Such measures give rise to a natural probability distribution on the set of lattices: by a random lattice, we mean a lattice picked from this distribution. Random lattices have the following property (see [1] for a proof): with overwhelming probability, the minima of a random n-dimensional lattice L are all asymptotically close to the Gaussian heuristic, that is, for all $1 \leq i \leq n$

$$\frac{\lambda_i(L)}{(\mathrm{vol}L)^{1/n}} \approx \frac{\Gamma(1+n/2)^{1/n}}{\sqrt{\pi}} \approx \sqrt{\frac{n}{2\pi e}}.$$

Many of our experiments use random lattices: by average case, we will mean running the algorithm on a random lattice. To generate random lattices, we use the provable method of [13], like [28].

Random bases. There is unfortunately no standard notion of random bases for a given lattice. By a random basis, we will mean a basis made of rather large vectors, chosen in a heuristic random way (see for instance [12]). Note that it is possible to sample lattice vectors in a sound way, as described by Klein [18] (see a refined analysis in [31,11]). And from any set of linearly independent lattice vectors, one can efficiently derive a basis whose vectors are not much longer (see for instance [2]).

2.2 Lattice Problems

The most famous lattice problem is the shortest vector problem (SVP): Given a basis of a lattice L, find a lattice vector whose norm is $\lambda_1(L)$. But SVP has several (easier) variants which are all important for applications:

– Hermite-SVP: Given a lattice L and an approximation factor $\alpha > 0$, find a non-zero lattice vector of norm $\leq \alpha \cdot (\mathrm{vol}L)^{1/n}$. The LLL algorithm [20] and its blockwise generalizations [36,8,10] are designed as polynomial-time Hermite-SVP algorithms. They achieve an approximation factor $(1+\varepsilon)^n$ exponential in the lattice dimension n where $\varepsilon > 0$ depends on the algorithm and its parameters. This exponential factor can actually be made slightly subexponential while keeping the running time polynomial.
– Approx-SVP: Given a lattice L and an approximation factor $\alpha \geq 1$, find a non-zero lattice vector of norm $\leq \alpha \cdot \lambda_1(L)$. Note that it might be difficult to verify a solution to this problem, since $\lambda_1(L)$ may not be known exactly. There are provably secure lattice-based cryptosystems [33,35] based on the worst-case quantum hardness of Approx-SVP with polynomial factor.
– Unique-SVP: Given a lattice L and a gap $\gamma > 1$ such that $\lambda_2(L)/\lambda_1(L) \geq \gamma$, find a shortest vector of L. There are cryptosystems [3,34] based on the worst-case hardness of Unique-SVP with polynomial gap: $n^{1.5}$ for [34] and n^7 for [3].

Any algorithm solving Approx-SVP with factor α also solves Hermite-SVP with factor $\alpha\sqrt{\gamma_n}$. Reciprocally, Lovász [21] showed that any algorithm solving Hermite-SVP with factor α can be used linearly many times to solve Approx-SVP with factor α^2 in polynomial time. There are also reductions [2,24] from the worst-case of Approx-SVP with a certain polynomial factor to the average-case (for a certain class of lattices) of Hermite-SVP with a certain polynomial factor.

Any algorithm solving Approx-SVP with factor α also solves Unique-SVP with gap $\geq \alpha$.

We will not discuss the closest vector problem (CVP), which is often used in cryptanalysis. However, in high dimension, the best method known to solve CVP heuristically transforms CVP into Unique-SVP (see for instance the experiments of [25]).

Knapsack lattices. An interesting class of lattices is the Lagarias-Odlyzko lattices [19] introduced to solve the knapsack problem: given n integers $x_1, \ldots, x_n$ uniformly distributed at random in $[1; M]$ and a sum $S = \sum_{i=1}^{n} \epsilon_i x_i$ where $\epsilon_i \in \{0, 1\}$ and $\sum \epsilon_i = \frac{n}{2}$, find all the ϵ_i. The Lagarias-Odlyzko (LO) lattice L [19] has the following property: if the density $d = n / \log_2(M)$ satisfies $d \leq 0.6463\ldots$, then with overwhelming probability, L has a unique shortest vector related to the ϵ_i, and $\lambda_1(L) \approx \sqrt{n/2}$. It has been proved [19] that there exists d_0 such that if $d \leq d_0/n$, then with overwhelming probability over the choice of the x_i's, L has exponential gap, which discloses the ϵ_i by application of LLL.

2.3 Lattice Algorithms

When the lattice dimension is sufficiently low, SVP can be solved exactly in practice using exhaustive search, thanks to enumeration techniques [37]. But beyond dimension 100, exhaustive search can be ruled out: only approximation algorithms can be run. Such algorithms try to output lattice bases $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ with small [illegible] $\|\mathbf{b}_1\| / \lambda_1(L)$, or small [illegible] $\|\mathbf{b}_1\| / \mathrm{vol}(L)^{1/n}$. The main approximation algorithms used in practice are the following:

LLL: it is a polynomial-time algorithm [20] which provably achieves (with appropriate reduction parameters) a Hermite factor $\lessapprox (4/3)^{(n-1)/4} \approx 1.075^n$ and an approximation factor $\lessapprox (4/3)^{(n-1)/2} \approx 1.154^n$, where n is the lattice dimension.

DEEP: the LLL algorithm with deep insertions [37] is a variant of LLL with potentially superexponential complexity. It is expected to improve the Hermite factor and the approximation factor of LLL, but no provable upper bound is known (except essentially that of LLL). The implementation of NTL actually depends on a blocksize parameter β: as β increases, one expects to improve the factors, and increase the running time.

BKZ: this is a blockwise generalization of LLL [37] with potentially superexponential complexity. The BKZ algorithm uses a blocksize parameter β: like DEEP, as β increases, one expects to improve the factors, and increase the running time. Schnorr [36] proved that if BKZ terminates, it achieves an approximation factor $\leq \gamma_\beta^{(n-1)/(\beta-1)}$. By using similar arguments as [36], it is not difficult to prove that it also achieves a Hermite factor $\leq \sqrt{\gamma_\beta}^{1+(n-1)/(\beta-1)}$.

DEEP and BKZ differ from the (theoretical) polynomial-time blockwise generalizations of LLL [36,8,10]: we will see that even the best polynomial-time algorithm known [10] seems to be outperformed in practice by DEEP and BKZ, though their complexity might be superexponential. The recent algorithm of [10] achieves a Hermite factor $\lessapprox \sqrt{\gamma_\beta}^{(n-1)/(\beta-1)}$ and an approximation factor $\lessapprox \gamma_\beta^{(n-\beta)/(\beta-1)}$.

Approximation algorithms exploit the triangular representation of lattice bases, related to orthogonalization techniques. Given a basis $B = [\mathbf{b}_1, ..., \mathbf{b}_n]$, the Gram-Schmidt orthogonalization (GSO) process constructs the unique pair (μ, B^*) of matrices such that $B = \mu B^*$ where μ is lower triangular with unit diagonal and $B^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ has orthogonal row vectors. If we represent the basis B with respect to the orthonormal basis $[\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \ldots, \mathbf{b}_n^*/\|\mathbf{b}_n^*\|]$, we obtain a triangular matrix whose diagonal coefficients are the $\|\mathbf{b}_i^*\|$'s. Thus, $\mathrm{vol}(B) = \prod_{i=1}^n \|\mathbf{b}_i^*\|$. LLL and BKZ try to limit the decrease of the diagonal coefficients $\|\mathbf{b}_i^*\|$.

It is sometimes useful to look at more than just the quality of the first basis vector $\mathbf{b}_1$. In order to evaluate the global quality of a basis, we define the Gram-Schmidt log (GSL) as the sequence of the logarithms of the $\|\mathbf{b}_i^*\|$: $\mathrm{GSL}(B) = \left(\log\left(\|\mathbf{b}_i^*\| / \mathrm{vol} L^{1/n}\right)\right)_{i=1..n}$. It is folklore that the GSL often looks like a decreasing straight line after running reduction algorithms. Then the average slope η of the GSL can be computed with the least mean squares method: $\eta = 12 \cdot (\sum i \cdot \mathrm{GSL}(B)_i) / ((n+1) \cdot n \cdot (n-1))$. When the GSL looks like a straight line, the Hermite factor H and the average slope η are related by $\log(H)/n \approx -\eta/2$.

3 Experimental Quality of Lattice Reduction Algorithms

In this section, we give a concrete picture of what lattice reduction algorithms can achieve today, and we compare it with the best theoretical results known. All our experiments were performed with the NTL 5.4.1 library [39].

First of all, we stress that SVP and its variants should all be considered easy when the lattice dimension is less than 70. Indeed, we will see in Section 4 that exhaustive search techniques [37] can solve SVP within an hour up to dimension 60. But because such techniques have exponential running time, even a 100-dimensional lattice is out of reach.

When the lattice dimension is beyond 100, only approximation algorithms like LLL, DEEP and BKZ can be run, and the goal of this section is to predict what they can exactly achieve. Before giving the experimental results, let us say a few words on the methodology. We have ran experiments on a large number of samples, so that an average behaviour can be reasonably conjectured. For each selected lattice, we ran experiments on at least twenty randomly chosen bases, to make sure that reduction algorithms did not take advantage of special properties of the input basis: the randomization must make sure that the basis vectors are not short. Note that one cannot just consider the Hermite normal form (HNF): for instance, the HNF of NTRU lattices has special properties (half of its vectors are short), which impacts the behaviour of lattice algorithms (see [9]). This means that we will ignore the effect of choosing special input bases: for instance, if one applies LLL on the standard basis of the LO lattice [19], or any permutation of its rows, it can be shown that if the density is d is lower bounded by $d_0 > 0$, then the first vector output by LLL approximates the shortest vector by a subexponential factor $2^{O(\sqrt{n})}$ rather than the general exponential factor $2^{O(n)}$. This phenomenon is due to the structure of orthogonal lattices [29].

Basis randomization allows to transform any deterministic algorithm like LLL or BKZ into a randomized algorithm. Experiments suggest that LLL and BKZ behave like probabilistic SVP-oracles in low dimension (see Fig. 1): no matter which lattice is selected, if the input basis is chosen at random, the algorithm seems to have a non-negligible probability of outputting the shortest vector.

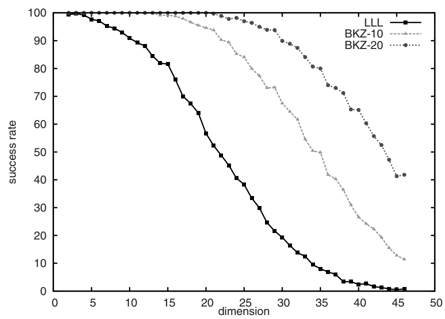


Fig. 1. Experimental probability of recovering the shortest vector, given a random basis of a random lattice, with respect to the dimension

3.1 Hermite-SVP

The Hermite factor achieved by reduction algorithms seems to be independent of the lattice, unless the lattice has an exceptional structure, in which case the Hermite factor can be smaller than usual (but not higher). By exceptional structure, we mean an unusually small first minimum $\lambda_1(L)$, or more generally, an unusually small Rankin invariant (that is, the existence of a sublattice of unusually small volume). In high dimension, we have never found a class of lattices for which the Hermite factor was substantially higher than for random lattices. We therefore speculate that the worst case matches the average case.

When the lattice has no exceptional structure, the Hermite factor of LLL, DEEP and BKZ seems to be exponential in the lattice dimension: Figure 2 shows the average Hermite factor, with respect to the lattice dimension and the reduction algorithm; and Figure 3 shows the logarithm of Figure 2. The figures show that the Hermite factor is approximately of the form e^{an+b} where n is the lattice dimension and (a, b) seems to only depend on the lattice reduction algorithm used. Since we are interested in rough estimations, we simplify e^{an+b} to c^n, and Figure 4 shows that a few samples are enough to have a reasonable approximation of c: indeed, when picking random bases of a given lattice, the distribution looks Gaussian. Figure 5 shows the evolution of c with respect to the lattice dimension and the reduction algorithm; the value c seems to converge

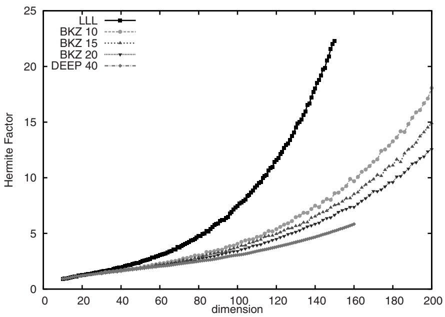


Fig. 2. The Hermite factor of LLL, BKZ and DEEP, depending on the dimension

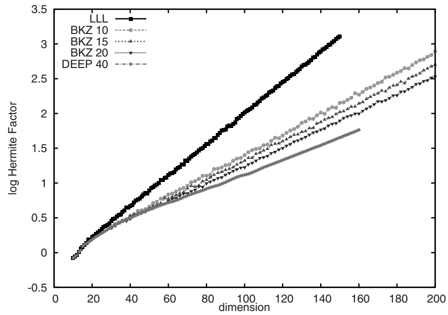


Fig. 3. Logarithm of Figure 2

Table 1. Average experimental Hermite factor constant of several approximation algorithms on random lattices, and comparison with theoretical upper bounds

	LLL	BKZ-20	BKZ-28	DEEP-50
$c =$ Hermite factor$^{1/n}$	1.0219	1.0128	1.0109	1.011
Best proved upper bound	1.0754	1.0337	1.0282	1.0754
η =average slope GSL	-0.0430	-0.0263	-0.0241	-0.026
Best proved lower bound	-0.1438	-0.0662	-0.0556	-0.1438

as the dimension increases. Table 1 gives the approximate value of c and the corresponding GSL slope η, depending on the algorithm, and compare it with the best theoretical upper bound known. It means that DEEP and BKZ have overall the same behaviour as LLL, except that they give much smaller constants, roughly the square root of that of LLL.

The case of LLL is interesting: it is well-known that the worst-case Hermite factor for LLL is $(4/3)^{(n-1)/4}$, reached by any lattice basis such that all its 2-dimensional projected lattices are critical. However, this corresponds to a worst-case basis, and not to a worst-case lattice. Indeed, when we selected such lattices but chose a random-looking basis, we obtained the same Hermite factor 1.02^n as with random lattices.

One can note that the constant c is always very close to 1, even for LLL, which implies that the Hermite factor is always small, unless the lattice dimension is huge. To give a concrete example, for a 300-dimensional lattice, we obtain roughly $1.0219^{300} \approx 665$ for LLL (which is much smaller than the upper bound $1.0754^{300} \approx 2176069287$) and $1.013^{300} \approx 48$ for BKZ-20 (which is much smaller than the upper bound $1.0337^{300} \approx 20814$). This implies that Hermite-SVP with factor n is easy up to dimension at least 450.

Figure 6 shows the evolution of the Hermite factor constant c for BKZ, as the blocksize increases, and provides two comparisons: one with the best theoretical upper bound known $\approx \sqrt{\gamma_\beta}^{1/(\beta-1)}$, using the best numerical upper bounds

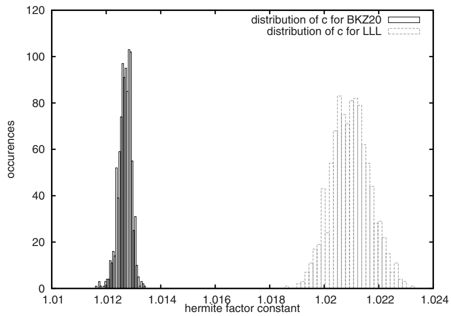


Fig. 4. Distribution of the Hermite factor constant, when picking random bases of a 160-dim lattice

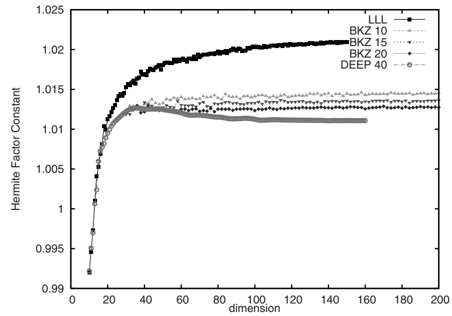


Fig. 5. Convergence of the Hermite factor constant c as the dimension increases

known on γ_β, and another with a prototype implementation of the best theoretical algorithm known [10], whose theoretical upper bound is $\sqrt{\gamma_\beta}^{1/(\beta-1)}$. We see that both BKZ and slide reduction [10] perform clearly much better than the theoretical upper bound, but BKZ seems better: slide reduction can be run with a much higher blocksize than BKZ, but even then, the constants seem a bit worse. The size of the gap between theory and practice is hard to explain: we do not have a good model for the distribution of the β-dimensional projected lattices used by BKZ; we only know that it does not correspond numerically to the distribution of a random lattice of dimension β. Figure 7 compares the Hermite factor constant c achieved by BKZ and DEEP, as the blocksize increases. It is normal that the constant achieved by BKZ is lower than DEEP for a fixed blocksize, since BKZ-reduced bases are also necessarily deep-reduced. But the comparison

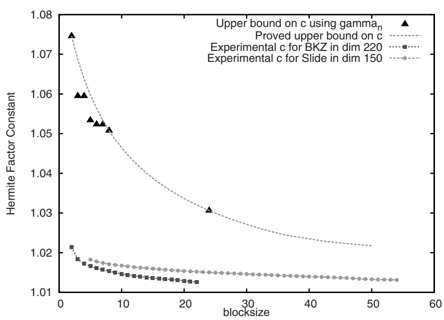


Fig. 6. Average value of the Hermite factor constant c for BKZ in high dimension, depending on the blocksize. Comparison with the best theoretical upper bound and with [10].

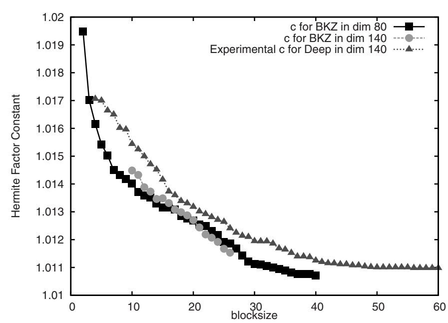


Fig. 7. Comparing the Hermite factor constant c for DEEP in high dimension and BKZ in dimension 80, depending on the blocksize

is important, because we will see in Section 4 that one can run DEEP on much bigger blocksize than BKZ, especially for high-dimensional lattices. This opens the possibility that DEEP might outperform BKZ for high-dimensional lattices. Figures 6 and 7 suggest that the best reduction algorithms known can achieve a Hermite factor of roughly 1.01^n in high dimension, but not much lower than that, since BKZ with very high blocksize is not realistic. For instance, a Hermite factor of 1.005^n in dimension 500 looks totally out of reach, unless the lattice has a truly exceptional structure.

3.2 Approx-SVP

As mentioned in Section 2, if we can solve Hermite-SVP with factor c^n in the worst case, then we can solve Approx-SVP with factor $\leq c^{2n}$. Thus, if we believe the previous experimental results on Hermite-SVP, we already expect the best reduction algorithms to solve in practice Approx-SVP with factor roughly $1.01^{2n} \approx 1.02^n$ in the worst case. More precisely, we can square all the values of Table 1 and Figures 6 and 7 to upper bound the approximation factor which can be achieved in practice. This means that Approx-SVP with factor n should be easy up to dimension at least 250, even in the worst case.

Surprisingly, we will see that one can often expect a constant much smaller than 1.02 in practice, depending on the type of lattices. First of all, as noticed in [28], the Hermite factor for random lattices is an upper bound for the approximation factor. More precisely, we know that for a random lattice, $\lambda_1(L)/\mathrm{vol}(L)^{1/n} \approx \frac{\Gamma(1+n/2)^{1/n}}{\sqrt{\pi}} \approx \sqrt{\frac{n}{2\pi e}}$, which means that if the Hermite factor is h, then the approximation factor is $\approx h/\sqrt{\frac{n}{2\pi e}}$. More generally, for any lattice L such that $\lambda_1(L) \geq \mathrm{vol}(L)^{1/n}$, the approximation factor is less than the Hermite factor: this means that on the average, we should achieve 1.01^n rather than 1.02^n. That would imply that Approx-SVP with factor n should be easy on the average up to dimension at least 500.

We have made further experiments to see if the worst case for Approx-SVP corresponds to the square of the Hermite factor, or something smaller. By the previous remark, the worst case can only happen for lattices L such that $\lambda_1(L) \leq \mathrm{vol}(L)^{1/n}$. But if $\lambda_1(L)$ becomes too small compared to $\mathrm{vol}(L)^{1/n}$, reduction algorithms might be able to exploit this exceptional structure to find the shortest vector. After testing various classes of lattices, the worst lattices for Approx-SVP which we have found are the following echelon lattices derived from the classical worst-case analysis of LLL. We call echelon basis a row matrix of the form:

$$\mathrm{Echelon}(\alpha) = \begin{bmatrix} \alpha^{n-1} & 0 & \dots & \dots & 0 \\ \alpha^{n-2}\cdot\sqrt{\alpha^2-1} & \alpha^{n-2} & \ddots & 0 & \vdots \\ 0 & \alpha^{n-3}\cdot\sqrt{\alpha^2-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \alpha & 0 \\ 0 & \dots & 0 & \sqrt{\alpha^2-1} & 1 \end{bmatrix}, \qquad (1)$$

where $\alpha \in [1; \sqrt{4/3}]$. It is easy to show that the reverse basis $C = (\mathbf{b}_n, \ldots, \mathbf{b}_1)$ is HKZ-reduced, and that the successive minima of the echelon lattice L satisfy: $\alpha^{k-1} < \lambda_k(L) \leq \alpha^k$, which allows to precisely estimate $\lambda_1(L)$. We have run the LLL algorithm on many echelon lattices (where the input basis is randomly chosen, not an echelon basis), depending on the value of α. The behaviour of LLL on such lattices is summarized by Figure 8. Two cases can occur:

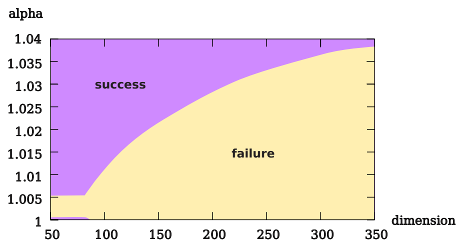


Fig. 8. Behaviour of LLL on echelon lattices, with respect to α and the dimension

- Either LLL succeeds in finding the shortest vector of the echelon lattice, in which case it actually finds the full HKZ-reduced basis. In particular, this happened whenever $\alpha > 1.043$,
- Either LLL fails to recover the shortest vector. Then the slope of the output GSL and the Hermite factor corresponds to those of random lattices: $c = 1.0219$ and $\eta = -0.043$. This means that the approximation factor of LLL is roughly α^n. Since α can be as high as 1.038 (in dimension 350) in Figure 8, this means that the approximation factor of LLL can be almost as high as the prediction $1.021^{2n} \approx 1.044^n$.

These experiments suggest that the worst case for Approx-SVP is very close to the square of the average Hermite factor for all reduction algorithms known, since this is the case for LLL, and the main difference between LLL and DEEP/BKZ is that they provide better constants. But the experiments also suggest that one needs to go to very high dimension to prevent reduction algorithms to take advantage of the lattice structure of such worst cases.

To summarize, it seems reasonable to assume that current algorithms should achieve in a reasonable time an approximation factor $\leq 1.01^n$ on the average, and $\leq 1.02^n$ in the worst case.

3.3 Unique-SVP

From a theoretical point of view, we know that if one can solve Approx-SVP with factor α in the worst-case, then we can solve Unique-SVP for all gap $\geq \alpha$. The previous section therefore suggests that we should be able to solve any Unique-SVP of gap roughly $\geq 1.02^n$, which corresponds to the square of the Hermite factor. In this section, we present experimental evidence which strongly suggest that Unique-SVP can be solved with a much smaller gap, namely a fraction of the Hermite factor 1.01^n, rather than the square of the Hermite factor. This means that Unique-SVP seems to be significantly easier than Approx-SVP.

The main difficulty with testing the hardness of Unique-SVP is to create lattices for which we precisely know the gap. We therefore performed experiments on various classes of lattices having a unique shortest vector.

Semi-Orthogonal Lattices. We first tested lattices for which the shortest vector was in some sense orthogonal to all other lattice vectors. More precisely, we chose lattices L for which the shortest vector $\mathbf{u}$ was such that $L' = L \cap \mathbf{u}^\perp$ was equal to the projection of L over $\mathbf{u}^\perp$: then $\lambda_2(L) = \lambda_1(L')$ and we chose L' in such a way that $\lambda_1(L')$ could be fixed, so as to select the gap of L. To be concrete, we tested the following two classes of lattices which are parameterized by a given pair (g_1, g_2) of real numbers. The two classes are

$$\begin{bmatrix} g_1 & 0 & \dots & 0 \\ 0 & g_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & g_2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} g_1 & 0 & 0 \dots 0 \\ 0 & M & 0 \dots 0 \\ 0 & r_1 & 1 \ddots \vdots \\ \vdots & \vdots & 0 \ddots 0 \\ 0 & r_{n-1} & 0 \;\; 0 \;\; 1 \end{bmatrix} \quad \text{where } r_i \in [1; M]$$

where M is a prime number, selected so that $\lambda_2(L) \approx g_2$: to do so, notice that the projection L' can be assumed to be random (see [13]), which gives a formula for $\lambda_1(L')$ depending simply on M.

Notice that the projected lattice L' is a hypercubic lattice for the first class, and a random lattice in the second class. In both cases, $(\mathrm{vol}L)^{1/n} / \lambda_1(L) \approx \lambda_2(L)/\lambda_1(L) \approx g_2/g_1$. The experiments on such lattices have been performed in dimensions 100 to 160, with g_2/g_1 between 2 and 20, and with randomly chosen bases.

For both classes, LLL is able to recover the unique shortest vector as soon as the gap is exponentially large, as shown by Figure 9. More precisely, for the first class, LLL recovers the unique shortest vector with high probability when the gap g_2/g_1 is a fraction of the Hermite factor, as shown by Figure 9 for instance $\geq 0.26 \cdot 1.021^n$ for the first class, and $\geq 0.45 \cdot 1.021^n$ for the second class. The smaller constants in the first class can perhaps be explained by the presence of an unusually orthogonal basis in the projected lattice, which triggers the success of LLL. Again, the behaviour of BKZ is similar to LLL, except that the constants are even smaller: in fact, the constants are so close to 1 that lattice dimensions $<$200 are too small to have good accuracy on the constants. For instance, BKZ-20 finds the shortest vector in dimension 200 in the first class, as soon as the gap is ≥ 2.09, and this limit grows up to 6.4 in dimension 300. This suggests that BKZ-20 retrieves the shortest vector when the gap is $\geq 0.18 \cdot 1.012^n$. Surprisingly, we will see in Section 5 that these very approximate BKZ-20 constants seem consistent with past high-dimensional experiments on the GGH challenges [12].

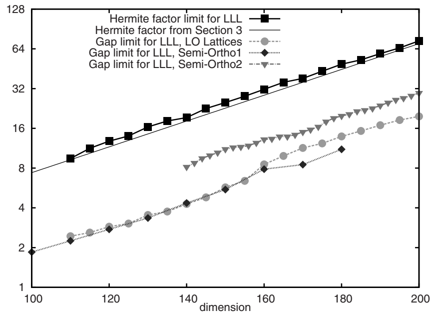


Fig. 9. Gap limits for solving Unique-SVP with LLL, and comparison with the Hermite factor

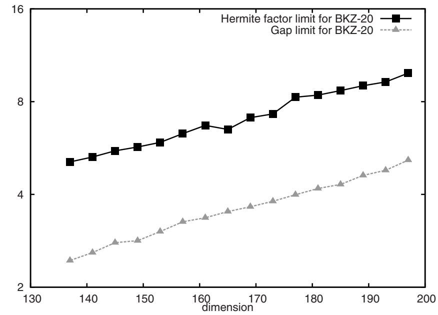


Fig. 10. Same as Figure 9, but with BKZ-20 on LO lattices

Knapsack lattices. The previous lattices have an exceptional structure compared to a general unique-SVP instance, which might bias the results. This suggests to test other types of lattices, such as the Lagarias-Odlyzko lattices [19]. In order to compare the results with those on semi-orthogonal lattices, we need to estimate the gap of LO lattices. Unfortunately, no provable formula is known for the second minimum of LO lattices. However, the analysis of Nguyen and Stern [30] suggests to heuristically estimate the gap from combinatorial quantities. More precisely, let $N(n, r)$ be the number of vectors in $\mathbb{Z}^n$ or norm $\leq \sqrt{r}$, which can easily be computed numerically. When r becomes large enough that $N(n, r) \gg M$, this hints that $\lambda_2(L) \approx \sqrt{r}$ (see [30]). It can be checked experimentally in low dimension that this heuristic approximation is very precise. As shown in Figures 9 and 10, the minimum gaps for which LLL or BKZ retrieve the shortest vector are once again proportional to the corresponding Hermite factors, that is in $0.25 \cdot 1.021^n$ for LLL and $0.48 \cdot 1.012^n$ for BKZ-20.

4 Running Times

In the previous section, we gave experimental estimates on the output quality of reduction algorithms. In this section, we now analyze the running-time growth to see if there are surprising phenomenons, and to guess what can be achieved in a reasonable time. We mainly ran the BKZ routine of NTL with quadratic precision to avoid floating-point issues, so the running times should not be considered as optimal.

4.1 Exhaustive Search

In low dimension, SVP can be solved exactly by exhaustive search: in practice, the most efficient method known is Schnorr-Euchner [37]'s enumeration, which is used as a subroutine in BKZ, and which outperforms the theoretical algorithms of Kannan [17] and AKS [4] (even though they have a much better theoretical complexity, see [31]). Given as input a reduced basis (the more reduced the basis, the faster the enumeration), it outputs the shortest vector in $2^{O(n^2)}$ polynomial-time operations. Figure 11 shows the average experimental running time of the enumeration (on a 1.7Ghz 64-bit processor), depending on the quality of the input basis (LLL, BKZ or DEEP). One can see that when the input basis is only LLL-reduced, the running time looks indeed superexponential $2^{O(n^2)}$. We also see that SVP can be solved in dimension 60 within an hour, but the growth of the curve also shows that a 100-dimensional lattice would take at least 35,000 years. A stronger preprocessing will reduce the curve a bit, but it is unlikely to make 100-dimensional lattices within reach.

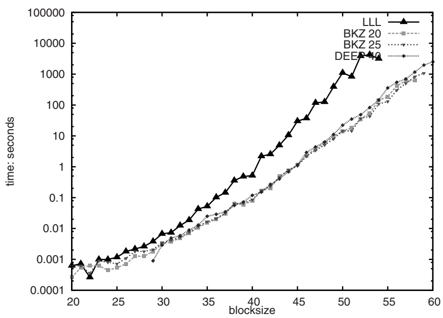


Fig. 11. Running time of the Schnorr-Euchner exhaustive search, depending on the preprocessing

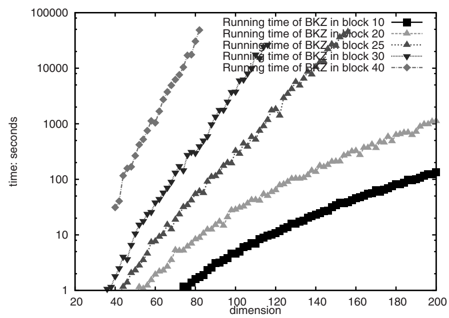


Fig. 12. Running time of BKZ in fixed blocksize

4.2 BKZ

No good upper bound on the complexity of BKZ and DEEP is known. If β is the blocksize and n is the lattice dimension, the best upper bound is $(n\beta)^n$ polynomial-time operations, which is super-exponential. But this upper bound does not seem tight: it only takes a few seconds to reduce a 100-dimensional lattice with blocksize 20. Since the theoretical analysis is not satisfying, it is very important to assess the experimental running time of BKZ, which is shown in Figures 13 and 12. Obviously, for fixed dimension, the running time of BKZ increases with the blocksize. But one can observe a brutal increase in the running time around blocksize 20 to 25 in high dimension, and the slope of the increase sharpens with the lattice dimension. We tried to determine the cause of this sudden increase. The increase does not seem to be caused by floating-point inaccuracies, as experiments with higher floating-point precision led to a similar phenomenon: Nor is it caused by the cost of the Schnorr-Euchner enumeration: exhaustive searches typically represent less than 1% of the total reduction time

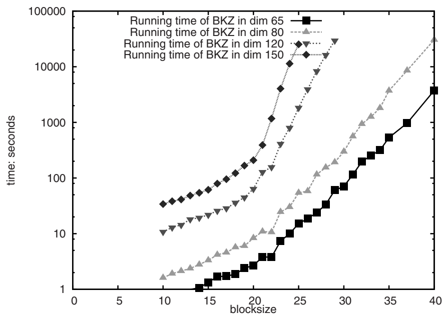


Fig. 13. Running time of BKZ in fixed dimension

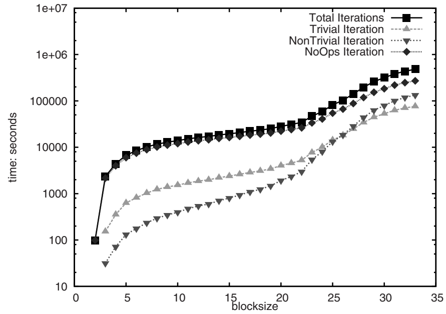


Fig. 14. Number of iterations in BKZ in dimension 100

in blocksize 25. In fact, it seems to be caused by a sudden increase in the number of calls to the Schnorr-Euchner enumeration. During a BKZ reduction, each exhaustive search inside a block gives rise to three possibilities:

1. Either the first block basis vector $\mathbf{b}_i^*$ is the shortest lattice vector in the block. Such cases are counted by `NoOps` in NTL.
2. Either the shortest lattice vector in the block is one of the β projected basis vectors. Such cases are counted by `Triv` in NTL.
3. Otherwise, the shortest lattice vector is neither of the β projected basis vectors. Then the algorithm has to do more operations than in the previous two cases. Such cases are counted by `NonTriv` in NTL.

After monitoring (see Figure 14), we observed that the `NoOps` case occurred most of the time, followed by `Triv` reductions and `NonTriv` reductions for blocksizes lower than 25. For higher blocksizes, `NoOps` was still the majority, but `NonTriv` iterations occurred more times than `Triv` iterations.

From Figures 13 and 12, we deduce that blocksizes much higher than 25 are not realistic in very high lattice dimension: the running time seems to be exponential in the dimension when the blocksize is ≥ 25, This is why we estimated the feasibility limit of the Hermite factor to roughly 1.01^n in Section 3, based on Figures 6 and 7: even if we were able to use blocksize 32, we would still not beat 1.01^n.

4.3 DEEP

Figure 15 gives the running time of the DEEP algorithm implemented in NTL, depending on the blocksize. Compared to Figure 13, we see that the running time of DEEP is much more regular than BKZ: there is no sharp increase at blocksize 20-25; the running time grows exponentially on a regular basis. Also the slope of the running-time of DEEP (in logarithmic scale) does not increase with the dimension of the lattice. This suggests that DEEP can be run in very

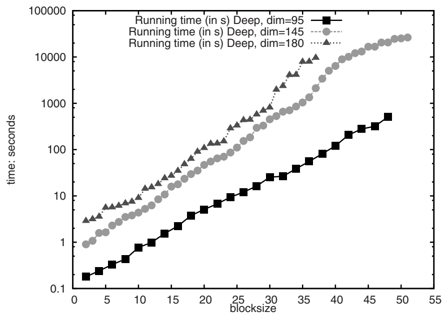


Fig. 15. Running time of DEEP in fixed dimension

high dimension with much higher blocksize than BKZ, which may make DEEP preferable to BKZ. However, Figure 7 showed that even with much higher blocksize, we do not expect to go significantly below the 1.01^n prediction for the Hermite factor.

5 Comparison with Former Lattice-Based Attacks

In Section 3, we tried to predict the asymptotical behaviour of the best reduction algorithms known. In this section, we compare our predictions with the largest lattice experiments ever done: surprisingly, our predictions seem consistent with the experiments, and may explain in retrospect why certain lattice attacks worked, but not others.

5.1 The GGH Challenges

In 1999, Nguyen [25] broke four GGH-challenges [12] in dimension 200, 250, 300 and 350, but the 400-dimensional challenge remained unbroken. The attack heuristically transformed a CVP-instance into a Unique-SVP instance, where a heuristic value for the gap of the Unique-SVP instance was known. The Unique-SVP instances arising from GGH-challenges look a bit like the first class of semi-orthogonal lattices: this is because GGH secret bases are slight perturbations of a multiple of the identity matrix.

By extrapolating the experimental results of Section 3, we can make a very rough guess of what should be the gap limit for which the BKZ-20 algorithm would solve the Unique-SVP instance corresponding to the GGH challenge. The results are given in Table 2. Even though the prediction $0.18 \cdot 1.012^n$ is only a rough estimate, the difference of magnitude shows that in retrospect, it was not a surprise that Nguyen [25] solved the GGH-challenges with BKZ-20 in dimension 200, 250 and 300. In dimension 350, the prediction is a bit worse, which is consistent with the fact that BKZ-20 failed: Nguyen [25] had to use a pruned BKZ-reduction to solve the GGH-350 challenge. In dimension 400, the

Table 2. Comparing predictions with past experiments on the GGH challenges

Dimension n	200	250	300	350	400
Estimation of the GGH gap	9.7	9.4	9.5	9.4	9.6
Gap estimate for BKZ-20 from Section 3	2.00	3.55	6.44	11.71	21.25
Algorithm used in [25]	BKZ-20	BKZ-20	BKZ-20	pruned-BKZ-60	Not broken

prediction is much worse than the expected gap, and it is therefore not a surprise that GGH-400 has not been solved. It seems that we would need much stronger reduction algorithms to solve GGH-400.

Recently, a weak instantiation of GGH was broken in [14], by solving Unique-SVP instances of polynomial gap using LLL up to at least dimension 1000. For many parameters, the numerical gap given in [14] is much lower than what could be hoped from our predictions for LLL, but there is a simple explanation. The problem considered in [14] is actually much easier than a general Unique-SVP problem: it is the embedding of a CVP problem when we already know a nearly-orthogonal basis and the target vector is very close to the lattice. This implies that LLL only performs a size-reduction of the last basis vector, which immediately discloses the solution. This also explains why the LLL running times of [14] were surprisingly low in high dimension.

Recently, a weak instantiation of GGH was broken in [14], by solving Unique-SVP instances of polynomial gap using LLL, up to at least dimension 1000. Surprisingly, for many parameters, the numerical gap of the instances solved in [14] is much lower than what could be hoped from our predictions for LLL. But there is an explanation. The problem considered in [14] is actually much easier than a general Unique-SVP problem: it is the embedding of a CVP problem when we already know a nearly- orthogonal basis and the target vector is very close to the lattice. This implies that LLL is fed with a special input basis (not a random basis), so special that LLL will only perform a size-reduction of the last basis vector, which will immediately disclose the solution. This explains why the LLL running times of [14] were surprisingly low in very high dimension. In other words, the attacks of [14] could even have been carried out without LLL.

5.2 The NTRU Lattices

The NTRU cryptosystem [15] is based on the hardness of lattice problems for the so-called NTRU lattices described in [15]. The key generation process of NTRU has changed several times over the past ten years: in the original article [15], the security was based on the hardness of SVP of the NTRU lattices, whereas more recent versions of NTRU are more based on the hardness of CVP in NTRU lattices. To simplify, we compare our predictions with the original description of NTRU based on SVP. In this case, NTRU lattices are essentially characterized by two parameters: N and q such that the dimension is $2N$, the volume is q^N, and there are heuristically N linearly independent shortest vectors of norm a bit smaller than $\sqrt{q}$ (and which are related to the secret key). Such lattices also have $2N$ trivial short vectors of norm q which are already known. Because NTRU lattices do not have a unique shortest vector, it is not clear if this fits any of the models of Section 3. But if we ever find the shortest vector, we will have found a non-zero vector smaller than q, which means solving Hermite-SVP for a suitable factor. Since we know the lattice volume, we can estimate the corresponding Hermite factor for all three historical NTRU parameter sets, as shown in Table 3. On the other hand, Section 3 suggests that we should be able

Table 3. Hermite factor required to solve the three historical NTRU parameter sets

Value of (N, q)	$(107, 64)$	$(167, 128)$	$(503, 256)$
Hermite factor required	$(1.00976)^{2N}$	$(1.00729)^{2N}$	$(1.00276)^{2N}$

to achieve a Hermite factor of roughly 1.01^{2N}: this means that out of the three NTRU parameter sets, only the first one $(N, q) = (107, 64)$ seems close to what can be achieved in a reasonable time. This parameter set was not supposed to be very secure (see [15]), but to our knowledge, no NTRU-107 lattice has ever been broken by direct lattice reduction. The only successful lattice attack was that of May in 1999 (see [22]), which combined exhaustive search with lattice reduction of smaller lattices. Surprisingly, it was estimated in [15] that NTRU-107 could be broken within a day using raw lattice reduction, but no actual break was reported: the experiments given in [15] only broke slightly smaller values of N. In fact, if we compute the Hermite factor corresponding to each NTRU instance broken in [15] using BKZ, similarly to Table 3, we obtain a Hermite factor of the form c^{2N} where c varies between 1.0116 and 1.0186: such values of c are clearly consistent the results of Section 3.

Still, since $(1.00976)^{2N}$ of Table 3 is very close to the prediction 1.01^{2N}, it seems reasonable to believe that NTRU-107 should be within reach of current algorithms, or small improvements. We therefore made experiments with three NTRU-107 lattices generated at random. Out of these three, only one was broken with BKZ: during the computation of BKZ-25, the shortest vector was found, but BKZ-25 did not even terminate. But BKZ did not succeed with the other lattices, and we stopped the computation after a few days. We then tested a stronger reduction algorithm on all three lattices, inspired by Figure 13:

– We partially reduce the NTRU-107 lattice with BKZ with increasing blocksize for a few hours.
– We project the lattice over the orthogonal complement of the first 107 vectors (we chose 107 based on the GSL slope): this gives a 107-dimensional projected lattice L' whose shortest vectors might be the projections of the initial 214-dimensional lattice L.
– We run BKZ on the projected lattice L' with increasing blocksize until an unusually short vector is found: because L' has much smaller dimension L, Figure 13 implies that we can run much higher blocksize. In practice, we

could reach blocksize 40. If the short vector is the projection of one of the shortest vectors of L, we can actually recover a shortest vector of L.

This experiment worked for all three NTRU-107 lattices: we were always able to recover the secret key, using BKZ of blocksize between 35 and 41 on the projected lattice, and the total running time was a few hours. By comparison, raw BKZ reduction only worked for one of the three lattices. This confirms that the Hermite factor prediction 1.01^n gives a good idea of what can be reached in practice. And knowing better the limits and the performances of current algorithms might help to design better ones.

Part of this work was supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT.

References

1. Ajtai, M.: Generating random lattices according to the invariant distribution (Draft of March 2006)
2. Ajtai, M.: Generating hard instances of lattice problems. In: Proc. of 28th STOC, pp. 99–108. ACM Press, New York (1996)
3. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Proc. of 29th STOC, pp. 284–293. ACM Press, New York (1997)
4. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proc. 33rd STOC, pp. 601–610. ACM Press, New York (2001)
5. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 1–11. Springer, Heidelberg (1999)
6. Cohn, H., Elkies, N.: New upper bounds on sphere packings. I. Ann. of Math (2) 157(2), 689–714 (2003)
7. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. of Cryptology 10(4), 233–260 (1997); Revised version of two articles from Eurocrypt 1996
8. Gama, N., Howgrave-Graham, N., Koy, H., Nguyen, P.Q.: Rankin's constant and blockwise lattice reduction. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 112–130. Springer, Heidelberg (2006)
9. Gama, N., Howgrave-Graham, N., Nguyen, P.Q.: Symplectic Lattice Reduction and NTRU. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 233–253. Springer, Heidelberg (2006)
10. Gama, N., Nguyen, P.Q.: Finding short lattice vectors within Mordell's inequality. In: STOC 2008: Proc. of the 40th Annual ACM Symposium on Theory of Computing, ACM Press, New York (2008)
11. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Report 2007/432 (2007) (to appear in STOC 2008), http://eprint.iacr.org/
12. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)

13. Goldstein, D., Mayer, A.: On the equidistribution of Hecke points. Forum Math 15(2), 165–189 (2003)
14. Han, D., Kim, M.-H., Yeom, Y.: Cryptanalysis of the Paeng-Jung-Ha cryptosystem from PKC 2003. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 107–117. Springer, Heidelberg (2007)
15. Hoffstein, J., Pipher, J., Silverman, J.: NTRU: a ring based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
16. Howgrave-Graham, N.A., Smart, N.P.: Lattice attacks on digital signature schemes. Des. Codes Cryptogr. 23(3), 283–290 (2001)
17. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proc. of 15th STOC, pp. 193–206. ACM Press, New York (1983)
18. Klein, P.: Finding the closest lattice vector when it's unusually close. In: Proc. of SODA 2000, ACM–SIAM (2000)
19. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. In: Journal of the Association for Computing Machinery (January 1985)
20. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Ann. 261, 513–534 (1982)
21. Lovász, L.: An Algorithmic Theory of Numbers, Graphs and Convexity. CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 50. SIAM Publications, Philadelphia (1986)
22. May, A., Silverman, J.H.: Dimension reduction methods for convolution modular lattices. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, Springer, Heidelberg (2001)
23. Micciancio, D., Goldwasser, S.: Complexity of lattice problems: A cryptographic perspective. In: The Kluwer International Series in Engineering and Computer Science, vol. 671, Kluwer Academic Publishers, Boston (2002)
24. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007) (electronic)
25. Nguyen, P.Q.: Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto 1997. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 288–304. Springer, Heidelberg (1999)
26. Nguyen, P.Q., Shparlinski, I.E.: The insecurity of the digital signature algorithm with partially known nonces. J. Cryptology 15(3), 151–176 (2002)
27. Nguyen, P.Q., Stehlé, D.: Floating-Point LLL Revisited. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)
28. Nguyen, P.Q., Stehlé, D.: LLL on the average. In: ANTS, pp. 238–256 (2006)
29. Nguyen, P.Q., Stern, J.: The two faces of lattices in cryptology. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, Springer, Heidelberg (2001)
30. Nguyen, P.Q., Stern, J.: Adapting density attacks to low-weight knapsacks. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 41–58. Springer, Heidelberg (2005)
31. Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. of Mathematical Cryptology (to appear, 2008)
32. Odlyzko, A.M.: The rise and fall of knapsack cryptosystems. In: Proc. of Cryptology and Computational Number Theory, Proc. of Symposia in Applied Mathematics, vol. 42, pp. 75–88. American Mathematical Society (1989)
33. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. Cryptology ePrint Archive, Report 2007/279 (2007) to appear in STOC 2008 http://eprint.iacr.org/

34. Regev, O.: New lattice-based cryptographic constructions. J. ACM 51(6), 899–942 (2004)
35. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 84–93. ACM Press, New York (2005)
36. Schnorr, C.-P.: A hierarchy of polynomial lattice basis reduction algorithms. Theoretical Computer Science 53, 201–224 (1987)
37. Schnorr, C.-P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Programming 66, 181–199 (1994)
38. Schnorr, C.-P., Hörner, H.H.: Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 1–12. Springer, Heidelberg (1995)
39. Shoup, V.: Number Theory C++ Library (NTL) version 5.4.1, http://www.shoup.net/ntl/

Efficient Sequential Aggregate Signed Data

Gregory Neven

Department of Electrical Engineering
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Heverlee-Leuven, Belgium
Gregory.Neven@esat.kuleuven.be
http://www.neven.org

Abstract. We generalize the concept of sequential aggregate signatures (SAS), proposed by Lysyanskaya, Micali, Reyzin, and Shacham (LMRS) at Eurocrypt 2004, to a new primitive called *sequential aggregate signed data* (SASD) that tries to minimize the total amount of transmitted data, rather than just signature length. We present SAS and SASD schemes that offer numerous advantages over the LMRS scheme. Most importantly, our schemes can be instantiated with *uncertified* claw-free permutations, thereby allowing implementations based on low-exponent RSA and factoring, and drastically reducing signing and verification costs. Our schemes support aggregation of signatures under keys of different lengths, and the SASD scheme even has as little as 160 bits of bandwidth overhead. Finally, we present a multi-signed data scheme that, when compared to the state-of-the-art multi-signature schemes, is the first scheme with non-interactive signature generation not based on pairings. All of our constructions are proved secure in the random oracle model based on families of claw-free permutations.

1 Introduction

Aggregate signatures (AS) [BGLS03] allow any third party to compress individual signatures $\sigma_1, \ldots, \sigma_n$ by n different signers on n different messages into an aggregate signature σ of roughly the same size as a single signature. Sequential aggregate signatures (SAS) [LMRS04] are a slightly restricted variant where the signers have to be organized in a sequence, each taking turns in adding their signature share onto the aggregate. Example applications of (S)AS schemes include secure routing protocols [KLS00], where routers authenticate paths in the network, and certificate chains in hierarchical public-key infrastructures, where certificate authorities (CA) authenticate public keys of lower-level CAs. Another important application area is that of battery-powered devices such as cell phones, PDAs, and wireless sensors that have to communicate over energy-consuming wireless channels.

Drawbacks of existing schemes. Only three instantiations of (S)AS schemes are presently known: the pairing-based $\mathcal{BGLS}$ [BGLS03] and $\mathcal{LOSSW}$ [LOS+06] schemes, and the $\mathcal{LMRS}$ [LMRS04] scheme based on families of certified [BY96]

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 52–69, 2008.

trapdoor permutations, but that with some tricks can be instantiated with RSA. All three schemes have some drawbacks though.

Pairings were only recently introduced to cryptography, and for the time being do not yet enjoy the same level of support in terms of standardization and implementations as for example RSA. The main disadvantage of the $\mathcal{LMRS}$ scheme on the other hand is that one of the tricks needed to turn RSA into a certified permutation is to use a verification exponent $e > N$.[1] This has a dramatic effect on the computational efficiency of signing and verification, because both require n long-exponent exponentiations for an aggregate signature containing n signatures.

Comparing this to pairing-based alternatives, the $\mathcal{BGLS}$ scheme also has rather expensive verification (n pairing computations), but at least has cheap signing (a single exponentiation). The $\mathcal{LOSSW}$ scheme has quite cheap signing and verification (two pairings and $160n$ multiplications), albeit at the price of only being secure in the weaker [illegible] (KOSK) model that requires signers to hand over (or at least prove knowledge of) their secret keys to a trusted CA. Both pairing-based schemes have shorter signatures than the $\mathcal{LMRS}$ scheme: 160 bits for $\mathcal{BGLS}$ and 320 bits for $\mathcal{LOSSW}$, versus 1024 bits for $\mathcal{LMRS}$ for a security level of 80 bits.

Finally, none of the existing schemes give the signers much freedom in choosing their own security parameters. This is particularly important for the certificate chain application, where a top-level CA probably wants higher-grade security than a private end-user. The pairing-based schemes require all signers to use the same elliptic-curve groups, so here the signers have no freedom whatsoever. A limited amount of freedom is allowed in the $\mathcal{LMRS}$ scheme, but signers have to be arranged according to [illegible] key size, which is exactly the opposite of what is needed for certificate chains.

Our contributions. We first observe that if one is truly concerned about saving bandwidth, then focusing solely on signature length is a bit arbitrary. Indeed, what really matters is the total amount of transmitted data, which contains messages, signatures, and in many applications the signers' public keys. (In fact, replacing the latter with shorter identity strings is the main motivation for identity-based aggregate signatures [GR06,BN07,BGOY07].) We therefore state our results in terms of a new, generalized primitive that we call [illegible] (SASD). The verification algorithm takes as only input the signed data Σ, and outputs vectors of public keys $\boldsymbol{pk} = ($[illegible]$_1, \ldots,$ [illegible]$_n)$ and messages $\boldsymbol{M} = (M_1, \ldots, M_n)$ to indicating that Σ correctly authenticates M_i under [illegible]$_i$ for $1 \leq i \leq n$, or $(\bot, \bot)$ to reject. The goal of the scheme is to keep the [illegible] to a minimum, i.e., the difference between the length of the signed data $|\Sigma|$ and that of the useful messages $\sum_{i=1}^{n} |M_i|$.

[1] Alternatively to choosing $e > N$, one could let each signer append to his public key a non-interactive zero-knowledge (NIZK) proof [BFM88] that $\gcd(e, \varphi(N)) = 1$. However, whether general NIZK proofs or special-purpose techniques [CM99,CPP07] are used, this invariably leads to a blowup in public key size and verification time, annihilating the gains of using aggregate signatures.

Table 1. Comparison of existing aggregate signature (AS), sequential aggregate signature (SAS), sequential aggregate signed data (SASD), multi-signature (MS), and multi-signed data (MSD) schemes. For each scheme we display whether its security relies on the knowledge of secret key (KOSK) or random oracle (RO) assumptions, on which number-theoretic assumptions it can be based (P for pairings, R for RSA, F for factoring), the net bandwidth overhead in bits, the cost of signing, and the cost of verification. Only the predominant terms are displayed. Symbols used are security parameters k_p, k_f, ℓ for pairings, factoring, and collision-resistance (typical values are $k_\mathrm{p} = \ell = 160$, $k_\mathrm{f} = 1024$); n for the number of signers; P for a pairing operation; E for a (multi-)exponentiation; and M for a multiplication. We give best/worst-case bounds for the overhead of the $\mathcal{SASD}$ and $\mathcal{MSD}$ schemes, as they depend on the length of the messages being signed.

Scheme	Type	KOSK	RO	Inst	Overhead	Sign	Vf
$\mathcal{BGLS}$ [BGLS03]	AS	N	Y	P	k_p	1E	nP
$\mathcal{LOSSW}$ [LOS+06]	SAS	Y	N	P	$2k_\mathrm{p}$	$2\mathrm{P} + n\ell\mathrm{M}$	$2\mathrm{P} + n\ell\mathrm{M}$
$\mathcal{LMRS}$ [LMRS04]	SAS	N	Y	R	k_f	nE	nE
$\mathcal{SASD}$	SASD	N	Y	R,F	$[\ell, k_\mathrm{f} + \ell]$	$1\mathrm{E} + 2n\mathrm{M}$	$2n$M
$\mathcal{SAS}$	SAS	N	Y	R,F	$k_\mathrm{f} + \ell$	$1\mathrm{E} + 2n\mathrm{M}$	$2n$M
$\mathcal{B}ol$ [Bol03]	MS	Y	Y	P	k_p	1E	$2\mathrm{P} + n\mathrm{M}$
$\mathcal{LOSSW}$ [LOS+06]	MS	Y	N	P	$2k_\mathrm{p}$	$2\mathrm{E} + \ell\mathrm{M}$	$2\mathrm{P} + (\ell + n)\mathrm{M}$
$\mathcal{MSD}$	MSD	N	Y	R,F	$[\ell, nk_\mathrm{f} + \ell]$	1E	$2n$M

We then present our main construction, the $\mathcal{SASD}$ scheme, based on families of trapdoor permutations in the random oracle model. Its main advantage over the $\mathcal{LMRS}$ scheme is that it does *not* require the permutations to be certified, thereby allowing much more efficient instantiations like low-exponent RSA, and the first instantiation ever from factoring. The construction itself can be seen as combining ideas from the $\mathcal{LMRS}$ scheme and the PSS-R signature scheme with message recovery [BR96]; the main technical contribution, we think, lies in the security proof, which requires complex "query bookkeeping" for the simulation to go through. The impact on efficiency is spectacular (see Table 1): verification takes $2n$ multiplications, signing takes one exponentiation and $2n$ multiplications, and this at a bandwidth overhead of only 160 bits, which until now was the exclusive privilege of pairing-based schemes. Moreover, the scheme allows signers to mix-and-match security parameters at will, allowing much more flexibility for use in the real world.

There is a small caveat here, namely that the promised overhead only holds if the messages being signed are of a (modest) minimum length. To show that our efficiency gains are not just due to the generalization of the primitive, we additionally present a "purebred" SAS scheme that has a typical overhead of 1184 bits, but that otherwise shares all the advantages offered by the $\mathcal{SASD}$ scheme.

Multi-signatures. A multi-signature (MS) scheme [IN83] is the natural equivalent of an (S)AS scheme where all signers authenticate the same message. The current state-of-the-art schemes based on RSA or factoring [BN06] have interactive signature generation; those based on pairings [Bol03,LOS+06] are only secure in the KOSK setting. The $\mathcal{BGLS}$ scheme could be seen as a MS scheme (taking into account the issues [BNN07] that arise when signing the same message), but has significantly less efficient verification.

Analogously to what we did for SASD schemes, we generalize the concept of MS schemes to *multi-signed data* (MSD) schemes. We present the $\mathcal{MSD}$ scheme that is the first RSA and factoring-based scheme with non-interactive signature generation, and that is the first efficient non-interactive scheme secure in the plain public-key setting, i.e. without making the KOSK assumption. Unlike the $\mathcal{SASD}$ scheme however, the bandwidth gains here are mainly due to message recovery effects, and disappear completely when very short messages are being signed.

2 Sequential Aggregate Signed Data

Notation. If $k \in \mathbb{N}$, then 0^k is the bit string containing k zeroes, and $\{0,1\}^k$ is the set of all k-bit strings. If x, y are bit strings, then $|x|$ denotes the length (in bits) of x, and $x\|y$ denotes a bit string from which x and y can be unambiguously reconstructed. If $k \in \mathbb{N}$, S is a set, and $y \in S$, then $\boldsymbol{x} = (x_1, \ldots, x_k) \in S^k$ is a k-dimensional vector, $\boldsymbol{x}\|y$ is the $(k+1)$-dimensional vector $(x_1, \ldots, x_k, y)$, and $\boldsymbol{x}|_i = (x_1, \ldots, x_i)$. Let ε and $\boldsymbol{\varepsilon}$ denote the empty string and the empty vector, respectively. If S is a set, then $x \xleftarrow{\$} S$ denotes the uniform selection of an element from S. If $\delta \in [0,1]$, then $b \xleftarrow{\delta} \{0,1\}$ denotes that b is assigned the outcome of a biased coin toss that returns 1 with probability δ and 0 with probability $1-\delta$. If A is a randomized algorithm, then $y \xleftarrow{\$} \mathsf{A}^{\mathsf{O}}(x)$ means that y is assigned the output of A on input x when given fresh coin tosses and access to oracle O.

Syntax. A *sequential aggregate signed data* (SASD) scheme is a tuple of three algorithms $\mathcal{SASD} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$. Each signer generates a key pair $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$ consisting of a public key pk and a secret key sk with security parameter $k \in \mathbb{N}$. The first signer in the sequence with key pair (pk_1, sk_1) creates the signed data Σ_1 for message M_1 by computing $\Sigma_1 \xleftarrow{\$} \mathsf{Sign}(sk_1, M_1)$. The n-th signer in the sequence receives from the $(n-1)$-st signer the aggregate signed data Σ_{n-1}, and adds his own signature on message M_n onto the aggregation by running $\Sigma_n \xleftarrow{\$} \mathsf{Sign}(sk_n, M_n, \Sigma_{n-1})$. He then sends Σ_n on to the $(n+1)$-st signer. The verifier checks the validity of Σ_n by running the verification algorithm $(\boldsymbol{pk}, \boldsymbol{M}) \leftarrow \mathsf{Vf}(\Sigma_n)$. This algorithm either returns lists of n public keys $\boldsymbol{pk}$ and messages $\boldsymbol{M}$, indicating that the signature correctly authenticates message M_i under public key pk_i for $1 \le i \le n$, or returns $(\perp, \perp)$ to indicate rejection. Correctness requires that the verification algorithm returns $(\boldsymbol{pk}, \boldsymbol{M})$ with probability one when the signed data is honestly generated by all signers as described above.

Security. We take our inspiration for the security notion of SASD from the unforgeability notion of SAS schemes [LMRS04,BNN07]. The game begins with the generation of the key pair $(pk^*, sk^*) \xleftarrow{\$} \mathsf{Kg}(1^k)$ of the honest user that will be targeted in the attack. The forger F is given $\boldsymbol{pk}^*$ as input and has access to a signing oracle $\mathsf{Sign}(sk^*, \cdot, \cdot)$. This oracle, on input a message M_n and aggregate signed data Σ_{n-1}, returns $\Sigma_n \stackrel{\$}{\leftarrow} \mathsf{Sign}(sk^*, M_n, \Sigma_{n-1})$. In the random oracle model [BR93], the forger is additionally given oracle access to one or more random functions.

At the end of its execution, F outputs its forgery Σ. The forger wins the game iff $\mathsf{Vf}(\Sigma) = (\boldsymbol{pk}, \boldsymbol{M}) \neq (\bot, \bot)$ and there exists an index $1 \leq i \leq |\boldsymbol{pk}|$ such that (1) $pk_i = pk^*$ and (2) F never made a signature query $\mathsf{Sign}(sk^*, M_i, \Sigma_{i-1})$ for any Σ_{i-1} such that $\mathsf{Vf}(\Sigma_{i-1}) = (\boldsymbol{pk}|_{i-1}, \boldsymbol{M}|_{i-1})$.

The advantage of F is the probability that it wins the above game, where the probability is taken over the coins of Kg, Sign, and F itself. In the random oracle model, the probability is also over the choice of the random function(s). We say that F $(t, q_{\mathrm{S}}, n_{\max}, \epsilon)$-breaks $\mathcal{SASD}$ if it runs in time at most t, makes at most q_{S} signature queries, and has advantage at least ϵ, and aggregates contain at most $n_{\max}$ signatures. This means that the aggregate signed data that F submits to the signing oracle can contain at most $n_{\max} - 1$ signatures, and that its forgery can contain at most $n_{\max}$ signatures. In the random oracle, we additionally bound the number of queries that the adversary makes to each random oracle separately.

3 Our Main Construction

Claw-free permutations. A family of claw-free trapdoor permutations Π consists of a randomized permutation generation algorithm Pg that on input 1^k outputs tuples (π, ρ, π^{-1}) describing permutations π, ρ over domain $D_\pi = D_\rho$ of size $|D_\pi| \geq 2^{k-1}$, and the corresponding trapdoor information for the inverse permutation π^{-1}. There must exist efficient algorithms that given π, x compute $\pi(x)$, that given ρ, x compute $\rho(x)$, and that given π^{-1}, x compute $\pi^{-1}(x)$ for any $x \in D_\pi$. Let t_π denote the time needed to compute $\pi(x)$. A claw-finding algorithm A is said to (t, ϵ)-break Π if it runs in time at most t and

$$\Pr\left[\, \pi(x) = \rho(y) \;:\; (\pi, \rho, \pi^{-1}) \stackrel{\$}{\leftarrow} \mathsf{Pg}(1^k)\,;\; (x, y) \stackrel{\$}{\leftarrow} \mathsf{A}(\pi, \rho) \,\right] \;\geq\; \epsilon\ .$$

Other ingredients. Let $k, \ell \in \mathbb{N}$ be security parameters, where ℓ is a system-wide parameter but k can be chosen by each signer independently as long as $k > \ell$. (Typical values for a security level of 80 bits in a factoring-based instantiation would be $k = 1024$ and $\ell = 160$.) Let Π be a family of claw-free trapdoor permutations so that associated to each permutation π in the family there exists an additive abelian group $\mathbb{G}_\pi \subseteq D_\pi$ such that $|\mathbb{G}_\pi| \geq 2^{k-1}$. Let $d = \min_{\pi \in \Pi}(|\mathbb{G}_\pi|/|D_\pi|)$ be the [illegible] of $\mathbb{G}_\pi$ in D_π. We stress that π need *not* be a permutation over $\mathbb{G}_\pi$, and that π need *not* be homomorphic with respect to the group operation in $\mathbb{G}_\pi$. Let $\mathsf{enc}_\pi : \{0,1\}^* \to \{0,1\}^* \times \mathbb{G}_\pi$ an efficient encoding algorithm that breaks up a message M into a (shorter) message m and an element $\mu \in \mathbb{G}_\pi$, and let $\mathsf{dec}_\pi : \{0,1\}^* \times \mathbb{G}_\pi \to \{0,1\}$ be the corresponding decoding algorithm that reconstructs M from (m, μ). We require that the decoding function is injective, meaning that $\mathsf{dec}_\pi(m, \mu) = \mathsf{dec}_\pi(m', \mu') \Rightarrow (m, \mu) = (m', \mu')$. Finally, let $\mathrm{H} : \{0,1\}^* \to \{0,1\}^\ell$ and $\mathrm{G}_\pi : \{0,1\}^\ell \to \mathbb{G}_\pi$ be public hash functions modeled as random oracles.

Intuition. Before presenting our $\mathcal{SASD}$ scheme, we provide some intuition into the construction. First consider the following signature scheme with message recovery, that could be seen as a non-randomized generalization of PSS-R [BR96]. The signer's public key is a permutation π, the secret key is π^{-1}. To sign a message M, he computes $(m, \mu) \leftarrow \mathsf{enc}_\pi(M)$, $h \leftarrow \mathrm{H}(M)$, and $X \leftarrow \pi^{-1}(\mathrm{G}_\pi(h) + \mu)$. The signature consists of the pair $\sigma = (X, h)$. Given partial message m and signature σ, a verifier recomputes $\mu \leftarrow \pi(X) - \mathrm{G}_\pi(h)$, $M \leftarrow \mathsf{dec}_\pi(m, \mu)$, and returns M iff $\mathrm{H}(M) = h$. Observe that if the encoding is sufficiently dense ($d \approx 1$), then the net signing overhead is limited to $|h| = \ell$ bits, since the bandwidth of X is reused entirely for message recovery.

Two observations lead from this scheme to our $\mathcal{SASD}$ scheme. First, the type of data that can be "embedded" in X is not restricted to parts of the signed message; it could also be used for example to embed the signature of the previous signer. (The same idea actually underlies the LMRS scheme.) Second, suppose the signer wants to add a second signature on M_2 on top of $\sigma_1 = (X_1, h_1)$. One idea to keep the net overhead at a constant ℓ bits could be to use $h_2 \leftarrow h_1 \oplus \mathrm{H}(M_2)$ and let the overall signed data be $(m_1, m_2, X_1, X_2, h_2)$. The verifier can then recover M_2 from (m_2, X_2, h_2); h_1 from (h_2, M_2); and M_1 from (m_1, X_1, h_1). He accepts iff $\mathrm{H}(M_1) = h_1$. A number of additional tweaks would be needed to make this scheme secure (we do not make any claims about its security here), but this is the rough idea.

The scheme. We associate to the above building blocks the $\mathcal{SAS}$ scheme as follows. Each signer generates permutations $(\pi, \rho, \pi^{-1}) \xleftarrow{\$} \mathsf{Pg}(1^k)$. The public key is , $\leftarrow \pi$, the secret signing key is , $\leftarrow \pi^{-1}$. The aggregate signing and verification algorithms are given below.

Algorithm $\mathsf{Sign}^{\mathrm{H,G}}(\pi\ \ ^1, M_n, \Sigma_{n\ \ 1})$:
- If $n = 1$ then $\Sigma_0 \leftarrow (\boldsymbol{\varepsilon}, \varepsilon, \varepsilon, 0^\ell)$
- Parse $\Sigma_{n\ \ 1}$ as $(\ \ , m_{n\ \ 1}, X_{n\ \ 1}, h_{n\ \ 1})$
- If $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{n\ \ 1}) = (\perp, \perp)$ then return $\perp$
- $(m_n, \mu_n) \leftarrow \mathsf{enc}_{\pi_n}(M_n \| m_{n\ \ 1} \| X_{n\ \ 1})$
- $h_n \leftarrow h_{n\ \ 1} \oplus \mathrm{H}(\ \ \|\pi_n, M_n, m_{n\ \ 1}, X_{n\ \ 1})$
- $g_n \leftarrow \mathrm{G}_{\pi_n}(h_n)$
- $X_n \leftarrow \pi_n{}^1(g_n + \mu_n)$
- Return $\Sigma_n \leftarrow (\ \ \|\pi_n, m_n, X_n, h_n)$

Algorithm $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma)$:
- Parse Σ as $(\ \ , m_n, X_n, h_n)$, $n = |\ \ |$
- For $i = n, \ldots, 1$ do
  - If $|\mathbb{G}_{\pi_i}| < 2^\ell$ then return $(\perp, \perp)$
  - $g_i \leftarrow \mathrm{G}_{\pi_i}(h_i)$; $\mu_i \leftarrow \pi_i(X_i) - g_i$
  - $M_i \| m_{i\ \ 1} \| X_{i\ \ 1} \leftarrow \mathsf{dec}_{\pi_i}(m_i, \mu_i)$
  - $h_{i\ \ 1} \leftarrow h_i \oplus \mathrm{H}(\ \ |_i, M_i, m_{i\ \ 1}, X_{i\ \ 1})$
- If $(m_0, X_0, h_0) = (\varepsilon, \varepsilon, 0^\ell)$
- Then return $(\ \ , \boldsymbol{M} = (M_1, \ldots, M_n))$
- Else return $(\perp, \perp)$.

Efficiency. Note that the verification algorithm only contains a simple check on the output size of $\mathrm{G}_{\pi_i}(\cdot)$, but does not check whether $\mathbb{G}_{\pi_i} \subseteq D_{\pi_i}$ or whether π_i describes a permutation over D_{π_i}. Indeed, unlike the LMRS scheme, the security analysis of our scheme points out that the security of an honest signer is affected by adversarially generated keys of cosigners, and thereby allows cheaper instantiations based on uncertified permutations such as low-exponent RSA and factoring. The real reason only becomes clear in the details of the security proof, but intuitively the difference is that in our scheme the data embedded in X_i, namely $M_i \| m_{i-1} \| X_{i-1}$, is passed as an extra argument to the hash function $\mathrm{H}(\cdot)$, as was done in the signature scheme with message recovery sketched above. The same trick cannot be applied to the LMRS scheme though because the embedded data (the previous signature) can only be recovered evaluating the hash function. Instead, Lysyanskaya et al. overcome this problem by simulating random oracles with range $\mathbb{G}_\pi$ by choosing $x \stackrel{\$}{\leftarrow} D_\pi$ and returning $\pi(x)$. For the simulation to be correct, they rely on the fact that even adversarially generated π are permutations. We refer to the security proof for more details.

Also note that signers can independently choose their own value of the security parameter k; for the system-wide parameter ℓ, a comfortably high value (e.g. $\ell = 256$ or even 512) can be agreed upon without too much impact on performance. The exact overall bandwidth overhead depends on the length of the signed messages, the efficiency of the encoding algorithm, the family of permutations being used, the signers' security parameters $k_1, \ldots, k_n$ and the density d. For typical instantiations however (see below) the net overhead varies from ℓ bits when sufficiently long messages are being signed (in particular $|M_i| \geq k_i - k_{i-1}$), up to $\ell + \max(k_1, \ldots, k_n)$ bits for short messages.

Finally, the list of public keys contained in the signed data can of course be omitted from the transmission if the verifier already knows them.

4 Instantiating Our Construction

Instantiations from RSA. An RSA key generator [RSA78] is a randomized algorithm $\mathsf{Kg}_{\mathrm{RSA}}$ that on input 1^k outputs tuples (N, e, d) where $N = pq$ is a k-bit product of two large primes and $ed = 1 \bmod \varphi(N)$. The RSA function $\pi(x) = x^e \bmod N$ is generally assumed to be a trapdoor one-way permutation over $D_\pi = \mathbb{Z}_N^*$, where d is the trapdoor that allows to compute $\pi^{-1}(x) = x^d \bmod N$. An algorithm A is said to (t, ϵ)-break the one-wayness of $\mathsf{Kg}_{\mathrm{RSA}}$ if it runs in time at most t and

$$\Pr\left[\, x^e = y \bmod N \;:\; (N, e, d) \stackrel{\$}{\leftarrow} \mathsf{Kg}_{\mathrm{RSA}} \,;\, y \stackrel{\$}{\leftarrow} \mathbb{Z}_N^* \,;\, x \stackrel{\$}{\leftarrow} \mathsf{A}(N, e, y) \,\right] \;\geq\; \epsilon \;.$$

One can associate a claw-free permutation family to $\mathsf{Kg}_{\mathrm{RSA}}$ by taking $\rho(x) = x^e \cdot y \bmod N$, where $y \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$. It is easy to see that if an algorithm A (t, ϵ)-breaks this claw-free permutation, then there exists an algorithm B that (t, ϵ)-breaks the one-wayness of $\mathsf{Kg}_{\mathrm{RSA}}$.

The most important advantage of our scheme over the LMRS scheme is that small verification exponents can be used, e.g. $e = 3$ or $e = 65537$, thereby reducing the cost of an exponentiation with exponent e to that of a couple of multiplications. Several options exist for the group $\mathbb{G}_\pi$, the additive group operation, the hash function $\mathrm{G}_\pi(\cdot)$, and the message encoding/decoding algorithms to be used. The most straightforward choice would be to use $\mathbb{G}_{N,e} = \mathbb{Z}_N^*$ with multiplication modulo N. A computationally more efficient choice however is to use $\mathbb{G}_{N,e} = \{0\|x \;:\; x \in \{0,1\}^{k-1}\}$ with the XOR operation. Alternatively, one can use the permutation family of [HOT04] to save one bit of bandwidth per signer, but this comes at the cost of doubling the verification time.

Instantiations from factoring. Let $\mathsf{Kg}_{\mathrm{Wil}}$ be a randomized algorithm that on input 1^k outputs tuples (N,p,q) where $N = pq$ is a k-bit product of primes p,q such that $p = 3 \bmod 8$ and $q = 7 \bmod 8$. For such integers N, also called Williams integers, we have that -1 is a non-square modulo N with Jacobi symbol $(-1|N) = +1$, and that 2 is a non-square with Jacobi symbol $(2|N) = -1$. Also, each square modulo N has four square roots (x_1,x_2,x_3,x_4) such that $x_1 = -x_2 \bmod N$, $x_3 = -x_4 \bmod N$, $(x_1|N) = (x_2|N) = +1$, and $(x_2|N) = (x_3|N) = -1$. Consider the permutation $\pi : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ defined as

$$\pi(x) \;=\; \begin{cases} x^2 & \bmod N \text{ if } (x|N) = +1 \text{ and } x < N/2 \\ -x^2 & \bmod N \text{ if } (x|N) = +1 \text{ and } x > N/2 \\ 2x^2 & \bmod N \text{ if } (x|N) = -1 \text{ and } x < N/2 \\ -2x^2 & \bmod N \text{ if } (x|N) = -1 \text{ and } x > N/2 \;. \end{cases}$$

Note that the Jacobi symbol $(x|N)$ can be computed in time $O(|N|^2)$ without knowing the factorization of N. The inverse permutation $\pi^{-1}(y)$ can be computed using trapdoor information p,q by finding $c \in \{1,-1,2,-2\}$ such that y/c is a quadratic residue modulo N and computing the four square roots (x_1,x_{-1},x_2,x_{-2}) of y/c modulo N, ordered such that $(x_1|N) = (x_{-1}|N) = +1$, $x_1 < x_{-1}$, $(x_2|N) = (x_{-2}|N) = -1$, $x_2 < x_{-2}$. The inverse of y is the root x_c. Since this is a permutation over $\mathbb{Z}_N^*$, the same group operations, hash functions and message encoding algorithms can be used as described for RSA above.

One can associate a family of claw-free trapdoor permutations to $\mathsf{Kg}_{\mathrm{Wil}}$ by taking $\rho(x) = \pi(x)\cdot r^2 \bmod N$ where $r \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$. Algorithm A is said to (t,ϵ)-factor $\mathsf{Kg}_{\mathrm{Wil}}$ if it runs in time at most t and

$$\Pr\Big[\, x \in \{p,q\} \;:\; (N,p,q) \stackrel{\$}{\leftarrow} \mathsf{Kg}_{\mathrm{Wil}} \,;\, x \stackrel{\$}{\leftarrow} \mathsf{A}(N) \,\Big] \;\geq\; \epsilon \;.$$

Given a claw $\pi(a) = \rho(b)$, one can see that $a/b \bmod N$ is a square root of r^2, which with probability 1/2 is different from $\pm r \bmod N$ and thereby reveals the factorization of N. Therefore, if an algorithm A (t,ϵ)-breaks the claw-free permutation, then there exists an algorithm B that $(t,\epsilon/2)$-factors $\mathsf{Kg}_{\mathrm{Wil}}$.

5 Security of Our Construction

We prove the security of the $\mathcal{SASD}$ scheme in the random oracle model under the claw-freeness of the permutation family Π. The following theorem gives a formal security statement with concrete security bounds.

Theorem 1. F $(t, q_S, q_H, q_G, n_{max}, \epsilon)$ *SASD* A (t', ϵ') Π

$$\epsilon' \geq \frac{\epsilon}{e(q_S+1)} - \frac{4\big(q_H + q_G + 2n_{max}(q_S+1)\big)^2}{2^\ell}$$

$$t' \leq t + (1/d + 2)\big(q_H + 2n_{max}(q_S+1) + n_{max}\big) \cdot t_\pi \ .$$

We prove Theorem 1 in two steps. First, we restrict our attention to a particular class of forgers that we call forgers, defined in Definition 1. In Lemma 1 we show that for any (non-sequential) forger F there exists a sequential forger S with about the same success probability and running time. Next, we show in Lemma 2 how a sequential forger can be used to find a claw in Π. The theorem then follows directly by combining Lemma 1 and Lemma 2.

Definition 1. S *SASD* sequential

H($\cdot$) G.($\cdot$) Sign($\cdot$,$\cdot$)
H($\cdot$) H($, M_n, m_{n-1}, X_{n-1})$ $n =$ | | $\leq n_{max}$ $|\mathbb{G}_{\pi_i}| \geq 2^\ell$ $1 \leq i \leq n$.
$\mathrm{H}(Q_n) = \mathrm{H}($ $, M_n, m_{n-1}, X_{n-1})$
$Q_1 = (\pi_1, M_1, \varepsilon, \varepsilon), Q_2 = ($ $|_2, M_2, m_1, X_1), \ldots, Q_{n-1} =$ ($|_{n-1}, M_{n-1}, m_{n-2}, X_{n-2})$ S H(Q_1), …, H(Q_{n-1})

$$\mathsf{dec}_{\pi_i}\big(m_i\ ,\ \pi_i(X_i) - \mathrm{G}_{\pi_i}(h_i)\big) \ = \ M_i \| m_{i-1} \| X_{i-1}\ ,$$

$h_i = h_{i-1} \oplus \mathrm{H}(Q_i)$ $1 \leq i \leq n$ $(m_0, X_0, h_0) = (\varepsilon, \varepsilon, 0^\ell)$

Sign($\pi^{-1}, M_n, \Sigma_{n-1}$)
$\Sigma_{n-1} = ($ $, m_{n-1}, X_{n-1}, h_{n-1})$ $n =$ | | $+ 1 \leq n_{max}$ $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{n-1}) \neq (\perp, \perp)$
H($, M_{n-1}, m_{n-2}, X_{n-2})$
$\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{n-1})$.
$\Sigma = ($ $, m_n, X_n, h_n)$ $n =$ | | $\leq n_{max}$ $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma) = ($ $, \boldsymbol{M}) \neq (\perp, \perp)$ $1 \leq i \leq n$ $\pi_i = \pi^*$ S Sign($\pi^{*-1}, M_i, \Sigma_{i-1}$) Σ_{i-1} $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{i-1}) = ($ $|_{i-1}, \boldsymbol{M}|_{i-1})$ $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{i-1})$

The following lemma shows that for any non-sequential forger F, there exists a sequential forger S with approximately the same success probability and running time as F.

Lemma 1. F $(t, q_S, q_H, q_G, n_{max}, \epsilon)$ *SASD* S $(t', q_S, q'_H, q'_G, n_{max}, \epsilon')$ *SASD*

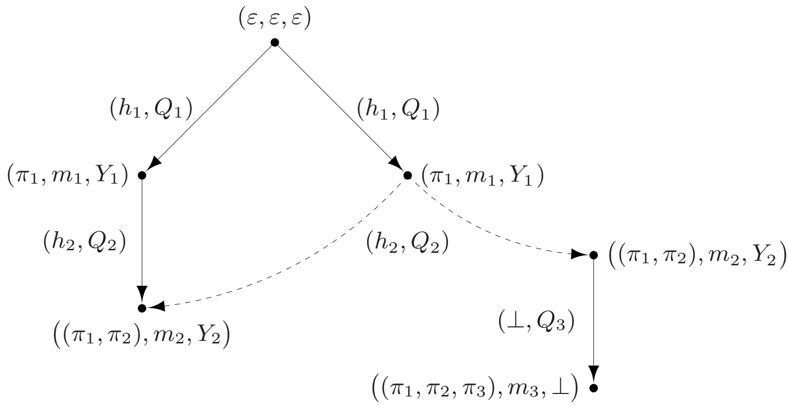


Fig. 1. The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ maintained by algorithm S. The solid edges indicate the state of $\mathcal{G}$ after F made sequential $\mathrm{H}(\cdot)$ queries $Q_1 = (\pi_1, M_1, \varepsilon, \varepsilon)$, $Q_2 = \big((\pi_1, \pi_2), M_2, m_1, X_1\big)$, $Q_1 = (\pi_1, M_1, \varepsilon, \varepsilon)$, and a non-sequential query $Q_3 = \big((\pi_1, \pi_2, \pi_3), M_3, m_2, X_2\big)$. The dashed edges depict the problematic cases when at that point F makes a new query $\mathrm{H}(Q_2) = \mathrm{H}\big((\pi_1, \pi_2), M_2, m_1, X_1\big)$ that causes event Bad to occur.

$$\begin{aligned} \epsilon' &\geq \epsilon - \frac{2\big(q_{\mathrm{H}} + q_{\mathrm{G}} + 2n_{\max}(q_{\mathrm{S}}+1)\big)^2}{2^\ell} \qquad (1) \\ q'_{\mathrm{H}} &\leq q_{\mathrm{H}} + n_{\max}(q_{\mathrm{S}}+1) \\ q'_{\mathrm{G}} &\leq q_{\mathrm{H}} + q_{\mathrm{G}} + 2n_{\max}(q_{\mathrm{S}}+1) \\ t' &\leq t + (q_{\mathrm{H}} + 2n_{\max}(q_{\mathrm{S}}+1)) \cdot t_\pi \ . \end{aligned}$$

Given a non-sequential forger F, we build a sequential forger S as follows. Given input π^* and access to oracles $\mathrm{H}'(\cdot)$, $\mathrm{G}'_\cdot(\cdot)$, and $\mathsf{Sign}'(\pi^{*-1}, \cdot, \cdot)$, algorithm S runs F on the same input π^* and simulates responses to F's $\mathrm{H}(\cdot)$, $\mathrm{G}_\cdot(\cdot)$ and $\mathsf{Sign}(\pi^{*-1}, \cdot, \cdot)$ oracle queries.

To satisfy Property 1 of Definition 1, S stores all previous responses to F's oracle queries in associative tables, retrieving the appropriate response from these tables when F asks the same query again. Note that the Sign algorithm is deterministic, so in a real attack repeating the same query to the signing oracle will result in the same signature being returned as well. Property 2 is satisfied by returning random values for F's "malformed" $\mathrm{H}(\cdot)$ queries. To answer F's $\mathrm{G}_\cdot(\cdot)$ queries, S simply relays responses from its own $\mathrm{G}'_\cdot(\cdot)$ oracle.

Correctly formed $\mathrm{H}(\cdot)$ queries are treated in a more complicated manner. S maintains a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as illustrated in Fig. 1. Each node is uniquely identified by a tuple $(\ , m, Y) \in \mathcal{V}$, and each edge is uniquely identified by a tuple $(h, Q) \in \mathcal{E}$. We explicitly allow multiple directed edges between the same pair of nodes. Initially, the graph only contains a so-called root node

$(\varepsilon, \varepsilon, \varepsilon)$. The idea is that all queries $\mathrm{H}(Q)$ satisfying Property 3, so-called queries, appear in edges in a tree rooted at $(\varepsilon, \varepsilon, \varepsilon)$, while all non-sequential queries appear in edges not connected to $(\varepsilon, \varepsilon, \varepsilon)$. We refer to the tree rooted at $(\varepsilon, \varepsilon, \varepsilon)$ as the . Property 3 is then enforced by letting S return $\mathrm{H}'(Q)$ for queries in the sequential tree, and random values for all other queries.

When F makes a new query $\mathrm{H}(Q_n) = \mathrm{H}(\, , M_n, m_{n-1}, X_{n-1})$, S adds a new edge to $\mathcal{G}$ as follows. If $n = 1$ and $m_0 = X_0 = \varepsilon$, then the query trivially satisfies Property 3, so it creates a new edge (h_1, Q_1) with the root as tail node and a new node $v_\mathrm{h} = (\pi_1, m_1, Y_1)$ as head node, where $(m_1, \mu_1) = \mathsf{enc}_{\pi_1}(M_1 \| \varepsilon \| \varepsilon)$, $h_1 = \mathrm{H}(Q_1)$, and $Y_1 = \mu_1 + \mathrm{G}_{\pi_1}(h_1)$. When $1 < n \leq n_{\max}$, algorithm S searches the graph for a node $v_\mathrm{t} = (\, |_{n-1}, m_{n-1}, Y_{n-1})$ where $Y_{n-1} = \pi_{n-1}(X_{n-1}))$. If such a node exists and it is in the sequential tree, then let (h_{n-1}, Q_{n-1}) be the incoming edge into v_t. We have that $\pi_{n-1}(X_{n-1}) = Y_{n-1} = \mu_{n-1} + \mathrm{G}_{\pi_{n-1}}(h_{n-1})$, so the requirements of Property 3 are satisfied by the sequence of queries $(Q_1, \ldots, Q_{n-1})$ on the path from the root to v_t. Algorithm S creates a new edge (h_n, Q_n) with tail node v_t and head node $v_\mathrm{h} = (\, , m_n, Y_n)$ where $h_n = h_{n-1} \oplus \mathrm{H}(Q_n)$, $(m_n, \mu_n) = \mathsf{enc}_{\pi_n}(M_n \| m_{n-1} \| X_{n-1})$, and $Y_n = \mu_n + \mathrm{G}_{\pi_n}(h_n)$. If v_t is not in the sequential tree, or if no such node v_t exists in the graph, then the query is deemed non-sequential. Algorithm S returns a random value as the random oracle response, and adds a new edge $(\perp, Q_n)$ to the graph with tail node $v_\mathrm{t} = (\, |_{n-1}, m_{n-1}, Y_{n-1})$ where $Y_n = \pi_{n-1}(X_{n-1})$ and head node $v_\mathrm{h} = (\, , m_n, \perp)$ where $(m_n, \mu_n) = \mathsf{enc}_{\pi_n}(M_n \| m_{n-1} \| X_{n-1})$, adding these nodes to the graph if they did not yet exist.

The creation of a new edge, however, should not violate the invariants that only sequential queries are represented by edges in the sequential tree, and that all of these queries were responded to using outputs of $\mathrm{H}'(\cdot)$. Two types of problems that can occur are illustrated by the dashed arrows for query Q'_2 in Fig. 1. The left arrow illustrates the situation when Q'_2 is such that the head node v_h of the new edge coincides with an existing node in the sequential tree. This is a problem, because if F later makes a query $\mathrm{H}(Q'_3)$ that "connects" to v_h, then there exist two different sequences (Q_1, Q_2) and (Q'_1, Q'_2) that satisfy the requirements of Property 3, violating the uniqueness requirement. The right dashed arrow in Fig. 1 illustrates the situation when v_h coincides with an existing node that is not part of the sequential tree. The newly created edge would suddenly incorporate v_h into the sequential tree, but this violates the invariant because S responded the query $\mathrm{H}(Q_3)$ with a random value, rather than with an output of $\mathrm{H}'(\cdot)$.

To preempt these problems, S aborts its execution whenever a new edge is added to the sequential tree with a head node that already exists in $\mathcal{G}$. We say that event Bad occurs when this happens.

If Σ_n are valid signed data, meaning $n \leq n_{\max}$ and $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_n) \neq (\perp, \perp)$, and event Bad does not occur, then all random oracle queries involved in the evaluation of $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_n)$ are sequential.

Let Σ_n be parsed as $(\, , m_n, X_n, h_n)$. We prove the claim by induction on the number of signatures n contained in Σ. The claim clearly holds for $n = 1$, because in this case the verification involves only a single query $\mathrm{H}(\pi_1, M_1, \varepsilon, \varepsilon)$ that is always sequential.

Suppose the claim is true for all signed data containing up to $n-1$ signatures. Let $Q_n, \ldots, Q_1$ be the $\mathrm{H}(\cdot)$ queries made when evaluating $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_n) = (\ \ , \boldsymbol{M})$, where $Q_i = (\ \ |_i, M_i, m_{i-1}, X_{i-1})$, and let $h'_1, \ldots, h'_{n-1}$ be the intermediate values obtained during the evaluation. If Σ_n is valid, then $\Sigma_{n-1} = (\ \ |_{n-1}, m_{n-1}, X_{n-1}, h'_{n-1})$ is also valid, namely $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{n-1}) = (\ \ |_{n-1}, \boldsymbol{M}|_{n-1})$. By the induction hypothesis, F must thus have made the queries $\mathrm{H}(Q_1), \ldots, \mathrm{H}(Q_{n-1})$ sequentially, so they must be represented in the graph $\mathcal{G}$ by edges $(h_1, Q_1), \ldots, (h_{n-1}, Q_{n-1})$ in the sequential tree. Clearly, we have that $h_i = h'_i = \mathrm{H}(Q_1) \oplus \ldots \mathrm{H}(Q_i)$ for $1 \leq i \leq n-1$, and that $h_n = h_{n-1} \oplus \mathrm{H}(Q_n)$.

Suppose for contradiction that F queries $\mathrm{H}(Q_n)$ non-sequentially, so it queries $\mathrm{H}(Q_n)$ at least before it queries $\mathrm{H}(Q_{n-1})$. At the moment that F queried $\mathrm{H}(Q_n)$, S created an edge $(\perp, Q_n)$ with tail node $\big(\ \ |_{n-1}, m_{n-1}, Y_{n-1} = \pi_{n-1}(X_{n-1})\big)$ not connected to the sequential tree. When F subsequently queries $\mathrm{H}(Q_{n-1})$, S adds edge (h_{n-1}, Q_{n-1}) to the sequential tree with head node $\big(\ \ |_{n-1}, m_{n-1}, Y'_{n-1} = \mu_{n-1} + \mathrm{G}_{\pi_{n-1}}(h_{n-1})\big)$. Since $\mathsf{dec}_{\pi_{n-1}}$ is injective however, there is only a single value of μ_{n-1} so that $\mathsf{dec}_{\pi_{n-1}}(m_{n-1}, \mu_{n-1}) = M_{n-1} \| m_{n-2} \| X_{n-2}$. In the verification of Σ_n, this value is recovered as $\mu_{n-1} = \pi_{n-1}(X_{n-1}) - \mathrm{G}_{\pi_{n-1}}(h_{n-1})$, so we have that $\mu_{n-1} = \pi_{n-1}(X_{n-1}) - \mathrm{G}_{\pi_{n-1}}(h_{n-1}) = Y'_{n-1} - \mathrm{G}_{\pi_{n-1}}(h_{n-1})$ and hence, because $\mathbb{G}_{\pi_{n-1}}$ is a group, that $Y_{n-1} = Y'_{n-1}$. This however means that the head node of (h_{n-1}, Q_{n-1}) coincides with the tail node of (h_n, Q_n), causing event BAD to occur. So if BAD does not occur, we conclude that F must query $\mathrm{H}(Q_n)$ after $\mathrm{H}(Q_{n-1})$, meaning sequentially. □

When F makes a signing query $\mathsf{Sign}(\pi^{*-1}, M_n, \Sigma_{n-1})$, algorithm S enforces Property 4 of Definition 1 by first verifying Σ_{n-1} and simulating an additional query $\mathrm{H}(\ \ \|\pi^*, M_n, m_{n-1}, X_{n-1})$. Only if Σ_{n-1} verifies correctly does S consult its own signing oracle; it relays the response of $\mathsf{Sign}'(\pi^{*-1}, M_n, \Sigma_{n-1})$ to F. Note that by Claim 5, if the event BAD does not occur, all $\mathrm{H}(\cdot)$ queries involved in the verification $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{n-1}) = (\ \ , \boldsymbol{M}) \neq (\perp, \perp)$ are sequential, so we also have that $\mathsf{Vf}^{\mathrm{H',G'}}(\Sigma_{n-1}) = (\ \ , \boldsymbol{M})$.

Finally, S ensures Property 5 by first verifying the forgery, simulating the necessary random oracle queries, and checking that the conditions with respect to the previous $\mathsf{Sign}(\pi^{*-1}, \cdot, \cdot)$ queries hold. Again, we rely here on Claim 5 to guarantee that if event BAD does not occur, then any valid signed data under random oracles $\mathrm{H}(\cdot), \mathrm{G}_{\cdot}(\cdot)$ is also valid under $\mathrm{H}'(\cdot), \mathrm{G}'_{\cdot}(\cdot)$.

It is clear that S is successful in breaking $\mathcal{SASD}$ whenever F is and event BAD does not occur. We want to bound the probability that BAD occurs. A detailed proof of the following claim, as well as a pseudo-code description of S and precise bounds on its running time and success probability, are provided in the full version [Nev08]. An intuition is given below.

[illegible] The probability that event BAD happens is at most

$$\Pr[\,\mathrm{BAD}\,] \;\leq\; \frac{2\big(q_{\mathrm{H}} + q_{\mathrm{G}} + 2n_{\max}(q_{\mathrm{S}}+1)\big)^2}{2^{\ell}} \;.$$

To see why the claim is true, observe that event BAD occurs when during the processing of a sequential query $\mathrm{H}(Q_n)$, the head node $(\ , m_n, Y_n = \mu_n + \mathrm{G}_{\pi_n}(h_n))$ of the created edge coincides with an existing node in the graph. At this point F's view is still independent of the value of $\mathrm{H}(Q_n)$, and therefore also of $h_n = h_{n-1} \oplus \mathrm{H}(Q_n)$. The probability that F previously queried $g_n = \mathrm{G}_{\pi_n}(h_n)$ is therefore at most its number of queries to $\mathrm{G.}(\cdot)$ divided by 2^ℓ. If it did not previously make this query, then g_n is a random value from D_{π_n}, and hence so is $Y_n = \mu_n + g_n$. Because for sequential queries we insisted that $|\mathbb{G}_{\pi_n}| \geq 2^\ell$, the probability that Y_n coincides with any of the existing nodes in $\mathcal{G}$ is at most the total number of nodes in the graph divided by 2^ℓ. Summing over all $\mathrm{H}(\cdot)$ queries yields the bound mentioned above. □

The next lemma shows that any sequential forger S can be turned into a claw-finding algorithm for Π. The proof reuses ideas from [BR96,Cor00,LMRS04].

Lemma 2. S $(t, q_\mathrm{S}, q_\mathrm{H}, q_\mathrm{G}, n_{\max}, \epsilon)$ *SASD* A (t', ϵ') Π

$$\epsilon' \geq \frac{\epsilon}{e(q_\mathrm{S}+1)} - \frac{q_\mathrm{H}(q_\mathrm{H}+q_\mathrm{G})}{2^\ell}$$

$$t' \leq t + ((1/d+1)q_\mathrm{H} + n_{\max}) \cdot t_\pi \ .$$

Given a sequential forger S against *SASD*, consider the following claw-finding algorithm A against Π. Algorithm A maintains initially empty associative arrays $[\cdot]$ and $[\cdot,\cdot]$. On input π^*, ρ^*, algorithm A runs S on target public key π^*, and responds to its oracle queries as follows:

Random oracle query $\mathrm{H}(Q_n)$**:** Parse Q_n as $(\ , M_n, m_{n-1}, X_{n-1})$. If $n > 1$, then A finds the unique sequence of queries $(Q_1, \ldots, Q_{n-1})$ as per Property 3 of a sequential forger, and looks up $[Q_{n-1}] = (c, x, h_{n-1})$. If $n = 1$, it sets $h_0 \leftarrow 0^\ell$.

If $\pi_n \neq \pi^*$ then A chooses $h \stackrel{\$}{\leftarrow} \{0,1\}^\ell$, computes $h_n \leftarrow h \oplus h_{n-1}$, stores $[Q_n] \leftarrow (\bot, \bot, h_n)$, and returns h to S.

If $\pi_n = \pi^*$ then A chooses $h \stackrel{\$}{\leftarrow} \{0,1\}^\ell$ and $c \stackrel{\delta}{\leftarrow} \{0,1\}$, and computes $(m_n, \mu_n) = \mathsf{enc}_\pi \ (M_n \| m_{n-1} \| X_{n-1})$ and $h_n \leftarrow h \oplus h_{n-1}$. If $c = 0$ then A repeatedly chooses $x \stackrel{\$}{\leftarrow} D_\pi$ and computes $g_n \leftarrow \pi^*(x) - \mu_n$ until $g_n \in \mathbb{G}_\pi$. If $c = 1$ then A repeatedly chooses $x \stackrel{\$}{\leftarrow} D_\pi$ and computes $g_n \leftarrow \rho^*(x) - \mu_n$ until $g_n \in \mathbb{G}_\pi$. (Each of these loops will require $1/d$ iterations on average.) If $[\pi^*, h_n]$ is already defined, then we say that event BAD_1 occurred and A aborts; otherwise, it sets $[\pi^*, h_n] \leftarrow g_n$. It stores $[Q_n] \leftarrow (c, x, h_n)$ and returns h to S.

Random oracle query $\mathrm{G}_\pi(h)$**:** If $[\pi, h]$ is not defined, then A randomly chooses $[\pi, h] \stackrel{\$}{\leftarrow} \mathbb{G}_\pi$. It returns $[\pi, h]$ to F.

Signing query $\mathsf{Sign}({\pi^*}^{-1}, M_n, \Sigma_{n-1})$: Parse Σ_{n-1} as $(\ , m_{n-1}, X_{n-1}, h_{n-1})$. Algorithm A looks up the entry $[\ \| \pi^*, M_n, m_{n-1}, X_{n-1}] = (c, X_n, h_n)$, which must exist by Property 4 of a sequential forger. Let $(m_n, \mu_n) = \mathsf{enc}_{\pi}\ (M_n \| m_{n-1} \| X_{n-1})$. If $c = 0$ then A returns $\Sigma_n = (\ \ \| \pi^*, m_n, X_n, h_n)$. If $c = 1$, then we say that event Bad_2 occurred and A aborts.

At the end of its execution, the forger outputs its forgery $\Sigma_n = (\ \ , m_n, X_n, h_n)$. By Property 5 the forgery is valid, so $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_n) = (\ \ , \boldsymbol{M})$ and there exists an index $1 \le i \le n$ such that $\pi_i = \pi^*$ and S never made a query $\mathsf{Sign}(\pi^{*-1}, M_i, \Sigma_{i-1})$ for the unique tuple Σ_{i-1} such that $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_{i-1}) = (\ \ |_{i-1}, \boldsymbol{M}|_{i-1})$. Let m_{i-1}, X_{i-1}, μ_i, and X_i be the intermediate values obtained during the computation of $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma_n)$.

Algorithm A looks up $\ \ [\ \ |_i, M_i, m_{i-1}, X_{i-1}] = (c, y, h_i)$ and $\ \ [\pi^*, h_i] = g_i$. (We know that these entries are defined by Property 5 of a sequential forger.) If $c = 0$ then we say that event Bad_2 occurred and A aborts. If $c = 1$ then we have that $\rho^*(y) = g_i + \mu_i$, but since Σ_n is valid we also have that $\pi^*(X_i) = g_i + \mu_i$. Since $\mathbb{G}_{\pi}$ is a group we therefore have that $\pi^*(X_i) = \rho^*(y)$; algorithm A outputs (X_i, y) as the claw for (π^*, ρ^*).

A detailed analysis in support of the bounds stated in the lemma reuses techniques due to Coron [Cor00], and is given in the full version [Nev08]. □

We can now shed some more technical light on how our security proof avoids relying on the family of permutations being certified like the LMRS scheme does. The LMRS scheme uses a full-domain random oracle, much like our $\mathrm{G}.(\cdot)$ oracle. In their proof, however, the responses of this oracle need to be simulated such that the claw-finding algorithm A knows a related preimage for permutations π. The usual trick of generating a random preimage $x \xleftarrow{\$} D_{\pi}$ and computing the output as $\pi(x)$ only gives the correct distribution if π is a permutation, hence the requirement that Π be certified. In our proof, the algorithm A only needs to know preimages related to queries $\mathrm{G}_{\pi}\ (\cdot)$, but not for queries $\mathrm{G}_{\pi}(\cdot)$ with $\pi \ne \pi^*$. It can therefore sample random elements from $\mathbb{G}_{\pi}$ directly.

6 Variations on the Main Construction

Sequential aggregate signatures. If for some reason the message recovery functionality is undesirable, then the following "purebred" sequential aggregate signature scheme $\mathcal{SAS}$ is easily derived from our $\mathcal{SASD}$ scheme. Just like the $\mathcal{SASD}$ scheme, the $\mathcal{SAS}$ scheme allows for efficient instantiations based on low-exponent RSA and factoring, and allows signers to independently choose their security parameter k. The signature size again depends on various issues such as the encoding efficiency and the permutation being used, but for signers with security parameters $k_1, \ldots, k_n$ will typically be about $\max(k_1, \ldots, k_n) + \ell$ bits.

The signer's public and private key are again a permutation π and its inverse π^{-1}; aggregate signing and verification are as follows:

Algorithm $\mathsf{Sign}^{\mathrm{H,G}}(\pi \ \ ^{1}, M_n, \ \ , \boldsymbol{M}, \sigma_{n \ \ 1})$:

If $n = | \ \ | = 1$ then $\sigma_0 \leftarrow (\varepsilon, \varepsilon, 0^{\ell})$
Parse $\sigma_{n \ \ 1}$ as $(x_{n \ \ 1}, X_{n \ \ 1}, h_{n \ \ 1})$
If $\mathsf{Vf}^{\mathrm{H,G}}(\ \ , \boldsymbol{M}, \sigma_{n \ \ 1}) = 0$ then return $\perp$
$(x_n, \xi_n) \leftarrow \mathsf{enc}_{\pi_n}(x_{n \ \ 1} \| X_{n \ \ 1})$
$h_n \leftarrow h_{n \ \ 1} \oplus$
$\quad \mathrm{H}(\ \ \| \pi_n, \boldsymbol{M} \| M_n, x_{n \ \ 1}, X_{n \ \ 1})$
$g_n \leftarrow \mathrm{G}_{\pi_n}(h_n)$
$X_n \leftarrow \pi_n^{\ \ 1}(g_n + \xi_n)$
Return $\sigma_n \leftarrow (x_n, X_n, h_n)$

Algorithm $\mathsf{Vf}^{\mathrm{H,G}}(\ \ , \boldsymbol{M}, \sigma_n)$:

Parse σ_n as (x_n, X_n, h_n), $n = | \ \ |$
For $i = n, \ldots, 1$ do
$\quad$ If $|\mathbb{G}_{\pi_i}| < 2^{\ell}$ then return 0
$\quad g_i \leftarrow \mathrm{G}_{\pi_i}(h_i)$; $\xi_i \leftarrow \pi_i(X_i) - g_i$
$\quad x_{i \ \ 1} \| X_{i \ \ 1} \leftarrow \mathsf{dec}_{\pi_i}(m_i, \mu_i)$
$\quad h_{i \ \ 1} \leftarrow h_i \oplus$
$\quad \quad \mathrm{H}(\ \ |_i, \boldsymbol{M}|_i, x_{i \ \ 1}, X_{i \ \ 1})$
If $(x_0, X_0, h_0) = (\varepsilon, \varepsilon, 0^{\ell})$ then return 1
Else return 0.

The scheme can be proved secure in the random oracle model under the notion of [LMRS04,BNN07]. The proof is almost identical to that of Theorem 1. The definition of a sequential forgers needs to be adapted to queries of the form $\mathrm{H}(\ \ , \boldsymbol{M}, x_{n-1}, X_{n-1})$, and nodes in the graph $\mathcal{G}$ will be identified by tuples $(\ \ , \boldsymbol{M}, x, Y)$. The concrete security bounds are identical to those obtained in Theorem 1.

Achieving tight security. Closer inspection of Theorem 1 learns that the reduction loses a factor q_{S} in the success probability of the claw-finding algorithm A. In principle, this means that higher security parameters have to be used in order to achieve the same security level, thereby increasing the length of keys and signatures. One can apply the techniques of Katz-Wang [KW03] however to obtain a scheme $\mathcal{SASD}t$ with a tight security reduction, at the minimal cost of an increase in signature length of n bits. (The same techniques have also been applied to achieve tight security for the LMRS scheme in [BNN07].) We refer to the full version [Nev08] for details.

7 Non-interactive Multi-signed Data

When all signers are authenticating the same message M, a more efficient scheme exists that does not require any interaction among the signers at all (as opposed to the sequential interaction required for the other schemes in this paper). Here, all signers independently generate their signature shares, which can then be combined by any third party into the final signature.

Syntax and security. A (MSD) scheme is a tuple of algorithms $\mathcal{MSD} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Comb}, \mathsf{Vf})$. A signer generates his own key pair via $(\ \ , \ \ ) \xleftarrow{\$} \mathsf{Kg}$. Each signer creates a partial signature on M via $\sigma \xleftarrow{\$} \mathsf{Sign}(\ \ , M)$. Any third party can combine a list of partial signatures into the final signed data via $\Sigma \xleftarrow{\$} \mathsf{Comb}(\boldsymbol{pk}, M, \ \ )$. The verification algorithm $\mathsf{Vf}(\Sigma)$ returns $(\boldsymbol{pk}, M)$ to indicate that Σ is valid for signers $\boldsymbol{pk}$ and message M, or returns $(\perp, \perp)$ to indicate rejection. Correctness requires that $\mathsf{Vf}(\Sigma) = (\ \ , M)$ if all signers behave honestly.

In the experiment defining security, the forger F is given a freshly generated $\ \ ^*$ as input, and has access to a signing oracle $\mathsf{Sign}(\ \ ^*, \cdot)$. It wins if it outputs a forgery Σ such that $\mathsf{Vf}(\Sigma) = (\boldsymbol{pk}, M) \neq (\perp, \perp)$ with $\ \ _i = \ \ ^*$ for some $1 \leq i \leq |\boldsymbol{pk}|$ and

F never queried M to the signing oracle. We say that F $(t, q_S, n_{\max}, \epsilon)$-breaks $\mathcal{MSD}$ if it runs in time at most t, makes at most q_S signing queries, its forgery contains at most $n_{\max}$ signatures, and wins the above game with probability at least ϵ. In the random oracle model, we additionally bound the maximum number of queries that F can make to each random oracle separately.

The scheme. Let $k, \ell \in \mathbb{N}$ be security parameters where k is chosen by each signer independently and ℓ is fixed system-wide. Let Π be a family of claw-free trapdoor permutations, let $\mathbb{G}_\pi \subseteq D_\pi$ for $\pi \in \Pi$ be a group, and let $\mathrm{H} : \{0,1\}^* \to \{0,1\}^\ell$ and $\mathrm{G}_\pi : \{0,1\}^\ell \to \mathbb{G}_\pi$ be random oracles, exactly as for the $\mathcal{SASD}$ scheme. The encoding and decoding functions are different though: we assume that $\mathsf{enc}_\pi : \{0,1\}^* \to \mathbb{G}_\pi$, $\mathsf{enc}_{(\pi_1,\ldots,\pi_n)} : \{0,1\}^* \to \{0,1\}^*$, and $\mathsf{dec}_{(\pi_1,\ldots,\pi_n)} : \{0,1\}^* \times \mathbb{G}_{\pi_1} \times \ldots \times \mathbb{G}_{\pi_n}$ are such that $\mathsf{enc}_{\pi_i}(M)$ outputs a group element $\mu_i \in \mathbb{G}_{\pi_i}$; $\mathsf{enc}_{\boldsymbol{\pi}}(M)$ outputs a partial message m; and the injective function $\mathsf{dec}_{\boldsymbol{\pi}}(m, \ )$ reconstructs the original message M. Key generation is consists again of generating a random permutation π as public key and its inverse π^{-1} as secret key; the other algorithms are described below.

Algorithm $\mathsf{Sign}^{\mathrm{H,G}}(\pi^{\ 1}, M)$:
- $\mu \leftarrow \mathsf{enc}_\pi(M)$; $h \leftarrow \mathrm{H}(M)$
- $g \leftarrow \mathrm{G}_\pi(h)$; $X \leftarrow \pi^{\ 1}(\mu + g)$
- Return X

Algorithm $\mathsf{Comb}^{\mathrm{H,G}}(\ , M, \boldsymbol{X})$:
- $m \leftarrow \mathsf{enc}_{\boldsymbol{\pi}}(M)$; $h \leftarrow \mathrm{H}(M)$
- Return $\Sigma \leftarrow (\ , m, \boldsymbol{X}, h)$

Algorithm $\mathsf{Vf}^{\mathrm{H,G}}(\Sigma)$:
- Parse Σ as $(\ , m, \boldsymbol{X}, h)$; $n \leftarrow |\ \ |$
- If $|\boldsymbol{X}| \neq n$ then return $(\perp, \perp)$
- For $i = 1, \ldots, n$ do
  - $g_i \leftarrow \mathrm{G}_{\pi_i}(h_i)$; $\mu_i \leftarrow \pi_i(X_i) - g_i$
- $M \leftarrow \mathsf{dec}_{\boldsymbol{\pi}}\big(m, (\mu_1, \ldots, \mu_n)\big)$
- If $\mathrm{H}(M) = h$ then return $(\ , M)$
- Else return $(\perp, \perp)$.

Security. The following theorem states that the $\mathcal{MSD}$ scheme is secure if the permutation family is claw-free. The proof uses techniques from [BR96, Cor00] and is provided in the full version [Nev08].

Theorem 2. [illegible] F [illegible] $(t, q_S, q_H, q_G, \epsilon)$ [illegible] $\mathcal{MSD}$ [illegible] A [illegible] (t', ϵ') [illegible] Π [illegible]

$$\epsilon' \geq \frac{\epsilon}{e(q_S + 1)} - \frac{(q_G + q_H + q_S + n_{\max} + 1)^2}{2^\ell}$$

$$t' \leq t + \frac{q_H + q_S + n_{\max} + 1}{d} \cdot t_\pi \ .$$

Instantiations. One can obtain instantiations of $\mathcal{MSD}$ from low-exponent RSA and factoring using the same permutation families and group structures described in Section 4. For the encoding function, one could for example split the message in $k_{\max}$-bit blocks (e.g. $k_{\max} = 4096$ when using RSA) and let μ be the first k bits of the block with index $\mathsf{h}(\pi, M)$ where $\mathsf{h} : \{0,1\}^* \to \{1,\ldots,\lfloor |M|/k_{\max} \rfloor\}$ is a non-cryptographic hash function. The function $\mathsf{enc}_{\boldsymbol{\pi}}(M)$ returns the remaining bits of M; decoding works by reconcatenating the different message parts in the correct order. For long enough messages M (in particular, $|M| \gg nk_{\max}$), there is no overlap between the message parts of different co-signers, and $\mathcal{MSD}$ achieves the promised length savings.

Alternatively, if the list of co-signers is known at the time of signing, one could modify the scheme so that encoding is more effective for short messages. Namely, one could use a single encoding function $\mathsf{enc}_{(\pi_1,\ldots,\pi_n)} : \{0,1\}^* \rightarrow \{0,1\}^* \times \mathbb{G}_{\pi_1} \times \ldots \times \mathbb{G}_{\pi_n}$ that ensures there is no overlap between the different message parts. In this case, however, the scheme needs to be modified to include in the computation of $h \leftarrow \mathrm{H}(\ , M)$, because otherwise there may exist (contrived) encoding algorithms that make the scheme insecure. Details are left as an exercise to the reader.

Acknowledgements

We thank the anonymous referees of EUROCRYPT 2008 for their helpful suggestions. The author is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO-Vlaanderen). This work was supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT, and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

BFM88. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: 20th ACM STOC, pp. 103–112. ACM Press, New York (1988)

BGLS03. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)

BGOY07. Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: ACM CCS 2007, pp. 276–285. ACM Press, New York (2007)

BN06. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: ACM CCS 2006, pp. 390–399. ACM Press, New York (2006)

BN07. Bellare, M., Neven, G.: Identity-based multi-signatures from RSA. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 145–162. Springer, Heidelberg (2006)

BNN07. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007)

Bol03. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)

BR93. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)

BR96. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)

BY96. Bellare, M., Yung, M.: Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. Journal of Cryptology 9(3), 149–166 (1996)

CM99. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)

Cor00. Coron, J.-S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)

CPP07. Catalano, D., Pointcheval, D., Pornin, T.: Trapdoor hard-to-invert group isomorphisms and their application to password-based authentication. Journal of Cryptology 20(1), 115–149 (2007)

GR06. Gentry, C., Ramzan, Z.: Identity-based aggregate signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257–273. Springer, Heidelberg (2006)

HOT04. Hayashi, R., Okamoto, T., Tanaka, K.: An RSA family of trap-door permutations with a common domain and its applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 291–304. Springer, Heidelberg (2004)

IN83. Itakura, K., Nakamura, K.: A public-key cryptosystem suitable for digital multisignatures. NEC Research & Development 71, 1–8 (1983)

KLS00. Kent, S., Lynn, C., Seo, K.: Secure border gateway protocol (S-BGP). IEEE JSAC 18(4), 582–592 (2000)

KW03. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM CCS 2003, pp. 155–164. ACM Press, New York (2003)

LMRS04. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)

LOS$^+$06. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)

Nev08. Neven, G.: Efficient sequential aggregate signed data. Cryptology ePrint Archive (2008)

RSA78. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signature and public-key cryptosystems. Communications of the Association for Computing Machinery 21(2), 120–126 (1978)

Proving Tight Security for Rabin–Williams Signatures

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. This paper proves "tight security in the random-oracle model relative to factorization" for the lowest-cost signature systems available today: every hash-generic signature-forging attack can be converted, with negligible loss of efficiency and effectiveness, into an algorithm to factor the public key. The most surprising system is the "fixed unstructured $B = 0$ Rabin–Williams" system, which has a tight security proof despite hashing unrandomized messages.

Table 1. Proven lower bounds on "security in the random-oracle model" relative to roots (for RSA) or factorization (for Rabin–Williams)

	B, number of bits of randomization of hash input		
	B large	$B = 1$	$B = 0$
Variable unstructured Rabin–Williams	tight security: '96 Bellare–Rogaway	no security: easy attack	no security: easy attack
Variable principal Rabin–Williams	tight security: this paper	loose security: this paper	loose security: this paper
Variable RSA	tight security: '96 Bellare–Rogaway	loose security: '93 Bellare–Rogaway	loose security: '93 Bellare–Rogaway
Fixed RSA	tight security: '96 Bellare–Rogaway	tight security: '03 Katz–Wang	loose security: '93 Bellare–Rogaway
Fixed principal Rabin–Williams	tight security: this paper	tight security: this paper	loose security: this paper
Fixed unstructured Rabin–Williams	tight security: '96 Bellare–Rogaway	tight security: this paper	tight security: this paper

* Permanent ID of this document: c30057d690a8fb42af6a5172b5da9006. Date of this document: 2008.01.30. This paper incorporates and supersedes various postings by the author between 2000 and 2007. This work was supported by the National Science Foundation under grants CCR–9983950, DMS–0140542, and ITR–0716498, and by the Alfred P. Sloan Foundation.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 70–87, 2008.

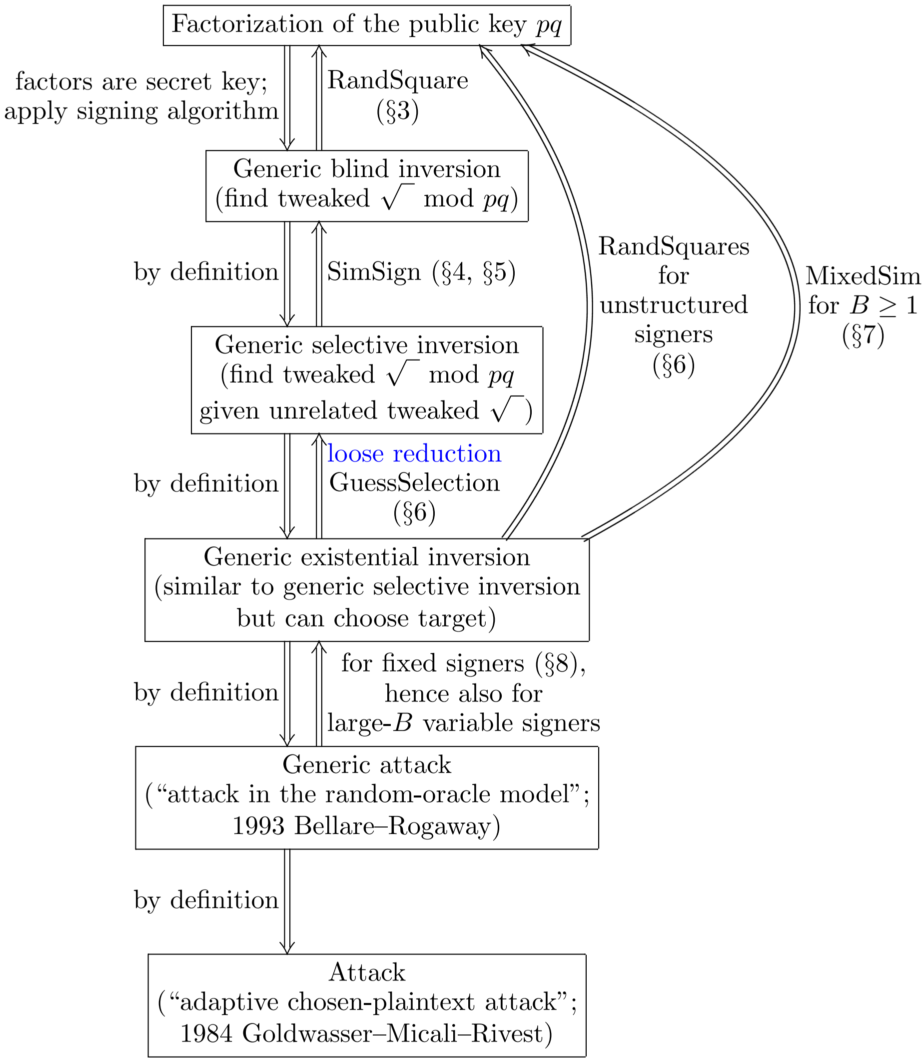


Fig. 1. Proven reductions among various types of attacks against Rabin- Williams signatures. SimSign is more difficult for Rabin-Williams than for RSA; the unstructured case was outlined in 1996 Bellare-Rogaway, but the principal case was specifically prohibited in 1996 Bellare-Rogaway and requires extra work performed in this paper. Guess Selection is standard but not tight. MixedSim is tight; it combines the new simulator with the central idea of 2003 Katz-Wang. Rand Squares is also new in this paper, and also tight; it can be viewed as the result of eliminating guesses from RandSquare (SimSign (GuessSelection))), or as the result of eliminating aborts from an overgeneralization of MixedSim.

Table 2. Summary of security-relevant parameters in the Rabin–Williams signature system. See Section 2 for definitions.

"K"	number of key bits; $0 < pq - 2^K < 2^K$
"D"	distribution of secret keys (p, q)
"H"	the hash function
"B"	number of bits of randomization of hash input
"α"	"unstructured": signer chooses uniform random tweaked $\sqrt{\ }$; "principal": signer finds unique tweaked $\sqrt{\ }$ that is a square etc.; "|principal|": if $\sqrt{\ }$ is between $(pq + 1)/2$ and $pq - 1$ then negate it
"β"	"variable": signer generates new random bits for each signature; "fixed": signer repeats signature if message is repeated

1 Introduction

Variants of the Rabin–Williams public-key signature system have, since 1980, held the speed records for signature verification. Are these systems secure?

There are many other signature systems of RSA/Rabin type. One can break each system by computing roots modulo the signer's public key pq or by breaking the system's hash function H. Are there other attacks?[1] This is not an idle concern: some RSA-type systems have been broken by devastating attacks that (1) are much faster than known methods to compute roots modulo pq and (2) work for a large fraction of all functions H, given oracle access to H.

Some systems have been proven immune to such attacks. For example, in the 1993 paper [5] that popularized this line of work (along with the terminology "secure in the random-oracle model"), Bellare and Rogaway proved the following security property for the traditional "FDH" form of exponent-e RSA: every H-generic attack on RSA-FDH can be converted (without serious loss of efficiency) into an algorithm to compute eth roots modulo pq.

Unfortunately, a closer look reveals that most of these proofs merely limit the devastation, without actually ruling it out. For example, the Bellare–Rogaway root-finding algorithm has only a $1/Q$ chance of success, where Q is the number of

[1] Notes on terminology: Twenty years ago, in [14, Section 2.2], Goldwasser, Micali, and Rivest defined various types of "attacks" against signature systems—in particular, "adaptive chosen-message attacks," the "most severe natural attack an enemy can mount." The definition has been repeated countless times in the literature, and the reader is assumed to be familiar with it. This paper follows common practice in abbreviating "adaptive chosen-message attack" as simply "attack."

This paper follows [5] in focusing on attacks that work for (a significant fraction of) all functions H, given access to an oracle computing H. In [5] these attacks are called attacks "in the random-oracle model." This paper follows the more concise terminology of [9, Section 7.1], [22, Section 1.1], [23, Section 4], et al.: these attacks are "H-generic attacks," or simply "generic attacks."

hash values seen by the FDH attack. Coron in [10] introduced a better algorithm having a $1/S$ chance of success, where S is the number of signatures seen by the FDH attack; but S can still be quite large.

Randomized signatures, in which B-bit random strings are prepended to messages before the messages are signed, allow much tighter proofs if B is large. For example, every H-generic attack on randomized exponent-e RSA (or Rabin's 1979 signature system) can be converted into an algorithm to compute eth roots modulo pq (or to factor pq) with a good chance of success. But generating random strings takes time, and transmitting the strings consumes bandwidth. Can we do better?

A 2002 theorem of Coron is widely interpreted as saying that FDH is stuck at $1/S$, i.e., that tight proofs require randomization of hash inputs; see [11]. A 2003 theorem of Katz and Wang allows much shorter random strings for some RSA variants but breaks down for Rabin–Williams. There are other systems with tight security proofs, but none of them offer state-of-the-art efficiency.

Contributions. This paper proves tight security for several state-of-the-art variants of the Rabin–Williams public-key signature system. What's most surprising is the "fixed unstructured $B = 0$" variant, a specific type of FDH that has a tight security proof despite hashing unrandomized messages. A minor technical assumption in Coron's theorem—the assumption of "unique" signatures—turns out to be a major loophole, producing a tight security proof from a random choice *later* in the Rabin–Williams signing process, after all hashing is done.

There are actually two security proofs in this paper. The "$B \geq 1$" proof uses a more general approach, pushing the Katz–Wang idea beyond the well-known "claw-free permutation pair" setting and carefully handling the "tweaked square roots" that appear in the Rabin–Williams system. The "unstructured $B = 0$" proof relies on a new proof idea that is more specific but also responsible for the aforementioned surprise. As far as I can tell, the new proof idea is tied to Rabin–Williams and cannot say anything useful about RSA; within the Rabin–Williams context, the new proof idea is tied to "unstructured" signers and does not cover "principal" or "|principal|" signers. The specific case of "fixed unstructured $B = 0$" Rabin–Williams is nevertheless worth studying because it is a state-of-the-art signature system of particular interest to implementors; among all high-speed systems with tight security proofs it is the only one that does not need to randomize hash inputs.

These proofs owe a heavy debt to the efforts of Koblitz and Menezes in [17] and [18] to clarify the limits of "provable security." In particular, in [17, Section 3.2], in the case of RSA with $B = 0$, Koblitz and Menezes explicitly stated an apparently new "RSA1" hard problem (which I call "generic existential inversion") and conjectured that it had the same difficulty as the usual hard problem for RSA (which I call "generic blind inversion"). The simplicity and clarity of the new hard problem inspired me to consider the analogous problem for Rabin–Williams. Koblitz and Menezes had commented that Coron's $1/S$ reduction could be translated to a $1/S$ reduction between these two hard problems, and that it was unreasonable to hope for a better reduction in light of Coron's

2002 theorem; I was quite surprised to discover that the "unstructured" case of the analogous Rabin–Williams conjecture could in fact be *proven*.

2 Parameters; Keys; Verification; Signing

This section defines the family of signature systems whose security is analyzed later in the paper. Standardizing a particular signature system in the family means standardizing various parameters: K, the number of key bits; D, the distribution of secret keys; H, the hash function; and B, the number of bits of randomization of the hash input. The signer's behavior is further controlled by two parameters relevant to security: first, a tweaked-square-root distribution α, either "unstructured" or "principal" or "|principal|"; second, a signature-repetition parameter β, either "fixed" or "variable." All of these parameters are explained in detail below.

Readers wondering "Why are you analyzing these specific systems?" should read the detailed cost analysis and historical survey in [8]. The short answer is that, among all the systems that are conjectured to provide a reasonable security level, these systems were engineered to minimize cost. (Exception: in applications where signature length is much more important than verification time, lower costs are achieved by systems of ElGamal–Schnorr–ECDSA type.) This engineering has not produced the world's simplest family of signature systems—this section needs two pages to state all the details of what the signer and verifier do—but the loss in simplicity is justified by the reduction in cost.

Secret keys and public keys. All users of the system know an integer $K \geq 10$. Typical choices of K include 1024 (not recommended), 1536, and 2048. All users of the system also know a distribution D (for example, the uniform distribution) of pairs of prime numbers (p, q) such that $p \in 3 + 8\mathbf{Z}$, $q \in 7 + 8\mathbf{Z}$, and $2^K < pq < 2^{K+1}$. Each signer chooses a random secret key (p, q) from the distribution D, and computes a corresponding public key pq.

For each algorithm A define PrFactor(A) as the probability that $A(pq) \in \{p, q\}$, when (p, q) is chosen randomly from the distribution D. This probability depends explicitly on A and implicitly on the parameters (K, D). No security is possible when K and D are chosen poorly. If $K = 512$, for example, then the attacker can use the number-field sieve to factor arbitrary integers between 2^K and 2^{K+1} with a moderate amount of effort, and can then freely forge signatures. As another example, if D has very little randomness and is concentrated on 2^{32} pairs (p, q), the attacker can factor pq by simply trying each of those 2^{32} pairs.

Theoreticians often simplify this picture by assuming that D is the uniform distribution. However, implementors often choose non-uniform distributions to save time in key generation. This paper considers arbitrary distributions of pairs (p, q), and thus arbitrary distributions of public keys pq; for each distribution D, this paper proves that various hard problems involving public keys from distribution D are equivalent to factoring public keys from distribution D.

Hashing and verification. All users of the system know an integer $B \geq 0$. Three interesting choices of B are 0, 1, and 128. All users of the system also know a function $H : \{B\text{-bit strings}\} \times \{\text{messages}\} \to \{1, 2, \ldots, 2^K\}$. For example, for $B = 0$ and $K = 2048$, the function H assigns an element of $\{1, 2, \ldots, 2^{2048}\}$ to each message. There are many popular choices of H, usually built from components such as MD5, SHA-1, and SHA-256.

A vector (e, f, s) is a **tweaked square root** of an integer h modulo a public key pq if $e \in \{1, -1\}$; $f \in \{1, 2\}$; $s \in \{0, 1, \ldots, pq - 1\}$; and $efs^2 \equiv h \pmod{pq}$. A vector (e, f, r, s) is a **signature** of a message m under a public key pq if r is a B-bit string and (e, f, s) is a tweaked square root of $H(r, m)$.

The difficulty of forging signatures depends on H. No security is possible when the hash function is chosen poorly. For example, if $H(r, m)$ is determined by MD5(m), then an attacker can find collisions in H by finding collisions in MD5.

Reader beware: Many authors allow the output range of H to be a function of the public key, but there cannot actually be any such dependence when H is a system parameter shared by all users, as it always is in practice. Putting a shared limit on the output range of H also means slightly changing the notion of a generic attack, and slightly changing the security proofs. My proofs include these minor changes.

Unstructured signers, principal signers, |principal| signers. Each message m has exactly 2^{B+2} signatures under pq: there are 2^B choices of r, and then 4 choices of tweaked square root (e, f, s) of $H(r, m)$ modulo pq. Which signature does the signer choose?

A stupid signer could easily expose his secret key to the attacker through this choice. For example, the signer could leak the ith bit of p in the ith signature as the bottom bit of r (if $B \geq 1$), as the Jacobi symbol of s modulo pq, etc. This example demonstrates that there is no hope of security if the signing function is chosen poorly. How do we know that a smarter-sounding signing algorithm does not have a similar leak?

There are three signature distributions proposed in the literature:

- **Unstructured:** The signer chooses a uniform random string r, and then a uniform random tweaked square root of $H(r, m)$, independently of all previous choices.
- **Principal:** The signer chooses a uniform random string r independently of all previous choices, and then chooses the principal tweaked square root of $h = H(r, m)$. This is the unique tweaked square root (e, f, s) such that e is 1 if h is a square modulo q, otherwise -1; f is 1 if eh is a square modulo p, otherwise 2; and s is a square modulo pq.
- **|Principal|:** The signer chooses a uniform random string r independently of all previous choices, and then chooses the "|principal|" tweaked square root of $H(r, m)$. If the principal tweaked square root is (e, f, s) then the |principal| tweaked square root is $(e, f, \min\{s, pq - s\})$; the point is that $\min\{s, pq - s\}$ takes a bit less space than s.

One step in this paper's security proofs—see Section 4—is split into three cases accordingly. A later step—see Section 6—is affected much more dramatically by the choice.

This paper is not the first paper to point out the importance of the signature distribution for Rabin–Williams security proofs. For example, Bellare and Rogaway in [7, Section 6] wrote "SignPRab ... returns a random square root ... We stress that a random root is chosen; a fixed root won't do." In my terminology, Bellare and Rogaway are requiring unstructured signers and prohibiting principal signers, |principal| signers, etc. Sometimes principal signers require extra work for a security proof (work done in Section 4 of this paper); sometimes they don't seem to allow a security proof at all.

Variable signers, fixed signers. What happens if the signer is given the same message to sign once again? There are two choices in the literature:

- **Fixed:** Given the same message again, the signer chooses the same signature again.
- **Variable:** Given the same message again, the signer generates a fresh signature, making random choices independently of the previous choices.

The importance of this choice for security proofs was first pointed out by Katz and Wang in [15]. The conventional wisdom before [15] was that tight security proofs required a large B; Katz and Wang proved tight security for various types of fixed signers with $B = 1$. As a more extreme illustration of the importance of this choice, consider the fact that "fixed unstructured $B = 0$" Rabin–Williams now has a tight security proof, whereas "variable unstructured $B = 0$" Rabin–Williams is easily breakable.

For principal and |principal| signatures with $B = 0$, no randomness is required, and variable signers are the same as fixed signers.

3 Generic Blind Inversion

Suppose we are given a public key pq and an integer $h' \in \{1, 2, \ldots, 2^K\}$. How quickly can we compute a tweaked square root of h' modulo pq? One approach is to factor pq; are there better approaches?

More formally: Fix K, D. For each algorithm A define PrInvBlind(A) as the probability that $A(pq, h')$ is some $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq - 1\}$ such that $e'f'(s')^2 \equiv h' \pmod{pq}$, when

- (p, q) is a D-distributed random secret key,
- h' is a uniform random element of $\{1, 2, \ldots, 2^K\}$,

and (p, q) is independent of h'. How large can PrInvBlind(A) be, as a function of the resources consumed by A?

Any fast probability-1 algorithm A for this generic-blind-inversion problem immediately implies a fast probability-1 algorithm to forge Rabin–Williams signatures, given oracle access to the hash function H. The attacker simply chooses the message m' that he wants to sign, chooses any B-bit string r', computes $h' = H(r', m')$, and uses A to compute a tweaked square root (e', f', s') of h'.

Then (e', f', r', s') is a signature of m'. Conversely, cryptanalysts trying to forge Rabin–Williams signatures will naturally consider this simple attack strategy as a first possibility.

Tight security proof. Unfortunately for the cryptanalyst, this problem is provably as difficult as factorization of public keys. Any fast high-probability algorithm A for this problem immediately implies a fast high-probability factorization algorithm RandSquare(A). The proof is completely standard, *except* for the details of how the tweaks e, f are handled; readers are encouraged to read the proof as a warmup for the security proofs in subsequent sections.

Here is the factorization algorithm RandSquare(A):

0. Input n.
1. Generate a uniform random vector $(e, f, s) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, n-1\}$.
2. Compute $h' = efs^2 \bmod n$.
3. Go back to step 1 if $h' \notin \{1, 2, \ldots, 2^K\}$.
4. Compute $(e', f', s') = A(n, h')$.
5. If $\gcd\{n, s' - s\} \notin \{1, n\}$, print it and stop.
6. If $\gcd\{n, s'\} \notin \{1, n\}$, print it and stop.

The following theorem states that a large success chance PrInvBlind(A) implies a similarly large factorization chance PrFactor(RandSquare(A)). The time of RandSquare(A) is practically identical to the time of A: the difference is a few easy operations modulo n to generate h, repeated only $n/2^K < 2$ times on average, plus a few gcd operations.

Theorem 3.1. PrFactor(RandSquare(A)) $\geq (1/2)$ PrInvBlind(A).

Proof. Let (p, q) be a D-distributed random secret key. The quantity $h' = efs^2 \bmod pq$ in step 4 of (RandSquare(A))(pq) is a uniform random element of $\{1, 2, \ldots, 2^K\}$; recall that each choice of h' is produced by exactly four choices of e, f, s. Thus the event $e'f'(s')^2 \equiv h' \pmod{pq}$ occurs with probability exactly PrInvBlind(A). I claim that, given this event, there is conditional probability at least $1/2$ that one of $s', s' - s$ has a nontrivial factor in common with pq.

Case 1: $\gcd\{h', pq\} = pq$. This is impossible, since $1 \leq h' \leq 2^K < pq$.

Case 2: $\gcd\{h', pq\} = p$. In this case $\gcd\{s', pq\} = p$ as desired.

Case 3: $\gcd\{h', pq\} = q$. In this case $\gcd\{s', pq\} = q$ as desired.

Case 4: $\gcd\{h', pq\} = 1$. I claim that $(s')^2 \equiv s^2 \pmod{pq}$. Notice first that $e'f'(s')^2 \equiv efs^2 \pmod{pq}$, and recall that p, q are primes with $p \in 3+8\mathbf{Z}$ and $q \in 7+8\mathbf{Z}$. Both possibilities for f, namely 1 and 2, are squares modulo q, so $f'(s')^2$ and fs^2 are squares modulo q, and both are nonzero since $\gcd\{h', q\} = 1$; the ratio e'/e is therefore a square modulo q and hence cannot be -1. Consequently $e' = e$ and $f'(s')^2 \equiv fs^2 \pmod{pq}$. Both $(s')^2$ and s^2 are squares modulo p, and both are nonzero since $\gcd\{h', p\} = 1$; the ratio f'/f is therefore a square modulo p and hence cannot be 2. Hence $f' = f$ and $(s')^2 \equiv s^2 \pmod{pq}$.

Recall that there are exactly four choices of e, f, s consistent with h', and observe that e', f', s' is independent of this choice. All four choices have the same e, f as I just showed, so only two of them have $s \equiv s'$ or $s \equiv -s'$. The other two choices occur with conditional probability 1/2; for those choices, pq divides $(s')^2 - s^2$ without dividing $s' - s$ or $s' + s$, so $\gcd\{n, s' - s\}$ is a nontrivial factor of pq. □

4 Generic Selective Inversion Using One Signature

Suppose we're given a public key pq, two integers $h, h' \in \{1, 2, \ldots, 2^K\}$, and a tweaked square root (e, f, s) of h modulo pq. How quickly can we compute a tweaked square root of h' modulo pq? One approach is to factor pq; are there better approaches?

More formally: Fix $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Also fix K and D. For each algorithm A define $\text{PrInvSelective}_1(A)$ as the probability that $A(pq, h, e, f, s, h')$ is some $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq - 1\}$ such that $e'f'(s')^2 \equiv h' \pmod{pq}$, when

- (p, q) is a D-distributed random secret key,
- h is a uniform random element of $\{1, 2, \ldots, 2^K\}$,
- (e, f, s) is an α-distributed random tweaked square root of h mod pq,
- h' is a uniform random element of $\{1, 2, \ldots, 2^K\}$,

and all of these choices are independent. How large can $\text{PrInvSelective}_1(A)$ be, as a function of the resources consumed by A?

This generic-selective-inversion problem is a natural step for the cryptanalyst beyond the generic-blind-inversion problem in Section 3. Any fast probability-1 algorithm A to solve this problem immediately implies a fast probability-1 algorithm to forge Rabin–Williams signatures, given oracle access to the hash function H. The forgery algorithm takes h and (e, f, s) from a legitimately signed message m, chooses a message $m' \neq m$, chooses a B-bit string r', computes $h' = H(r', m')$, computes $(e', f', s') = A(pq, h, e, f, s, h')$, and outputs (e', f', r', s') as a successful forgery of m'.

Similar comments apply to the problems articulated in subsequent sections. Each problem is a natural problem for the cryptanalyst to consider, providing more flexibility than the previous problem and potentially making attacks easier.

Tight security proof. Unfortunately for the cryptanalyst, this problem is provably as difficult as factorization of public keys. Any fast high-probability algorithm A for this problem immediately implies a fast high-probability algorithm $\text{SimSign}_1(A)$ for the generic-blind-inversion problem, and therefore implies a fast high-probability factorization algorithm $\text{RandSquare}(\text{SimSign}_1(A))$.

The intuition here is that A learns nothing from seeing h, e, f, s. It is well known how to formalize this intuition: namely, build a **simulator** that, given pq, generates (h, e, f, s) with exactly the same distribution as a signer who first generates h and then uses p, q to generate (e, f, s).

There are three different constructions of the simulator, and thus three different constructions of $\text{SimSign}_1(A)$, one for each of the three choices of α. Here is $\text{SimSign}_1(A)$ for the simplest choice, $\alpha =$ unstructured:

0. Input n and h'.
1. Generate a uniform random vector $(e, f, s) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, n-1\}$.
2. Compute $h = efs^2 \bmod n$.
3. Go back to step 1 if $h \notin \{1, 2, \ldots, 2^K\}$.
4. Print $A(n, h, e, f, s, h')$.

Here is $\text{SimSign}_1(A)$ for $\alpha \in \{\text{principal}, |\text{principal}|\}$:

0. Input n and h'.
1. Generate a uniform random $(e', f', x) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, n-1\}$.
2. Compute $g = \gcd\{x, n\}$.
3. If $g = n$ or $g \bmod 8 = 7$, set $e = 1$; otherwise set $e = e'$.
4. If $g = n$ or $g \bmod 8 = 3$, set $f = 1$; otherwise set $f = f'$.
5. Compute $s = x^2 \bmod n$.
6. Compute $h = efs^2 \bmod n$.
7. Go back to step 1 if $h \notin \{1, 2, \ldots, 2^K\}$.
8. Print $A(n, h, e, f, s, h')$ if $\alpha =$ principal, else $A(n, h, e, f, \min\{s, n-s\}, h')$.

The following theorem states that a large success chance $\text{PrInvSelective}_1(A)$ implies a large success chance $\text{PrInvBlind}(\text{SimSign}_1(A))$. The time of $\text{SimSign}_1(A)$ is practically identical to the time of A: the only difference is a few easy operations modulo n to generate h, repeated only $n/2^K < 2$ times on average.

Theorem 4.1. $\text{PrInvBlind}(\text{SimSign}_1(A)) = \text{PrInvSelective}_1(A)$.

The reader may have noticed that my constructions of $\text{SimSign}_1(A)$, in the principal and $|\text{principal}|$ cases, go to some extra work to handle extremely rare events such as $g = n$. The reward for this work is a particularly clean theorem. The simulators produce *exactly* the right output distribution, rather than producing *almost exactly* the right output distribution and forcing the user to worry about the difference.

Proof. Let (p, q) be a D-distributed random secret key. Write $n = pq$. Let h' be a uniform random element of $\{1, 2, \ldots, 2^K\}$, independent of (p, q). Consider $(\text{SimSign}_1(A))(n, h')$.

Unstructured: There are exactly four choices of (e, f, s) for each possible h; so the distribution of h is uniform, and (e, f, s) is a uniform random tweaked square root of h. Thus $e'f'(s')^2 \equiv h'$ with probability exactly $\text{PrInvSelective}_1(A)$.

Principal: If $e = 1$ then $h \equiv efs^2 = fs^2$ is a square modulo q since 2 is a square modulo q. If $e = -1$ then $h \equiv efs^2 = -fs^2$, which I claim is a non-square modulo q; otherwise q divides s, so q divides x, so $g = \gcd\{x, n\} \in \{n, q\}$, so $g = n$ or $g \bmod 8 = 7$, so $e = 1$, contradiction. Similarly, if $f = 1$ then $eh \equiv s^2$

is a square modulo p, and if $f = 2$ then $eh \equiv 2s^2$, which I claim is a non-square modulo p; otherwise p divides s, so p divides x, so $g = \gcd\{x, n\} \in \{n, p\}$, so $g = n$ or $g \bmod 8 = 3$, so $f = 1$, contradiction. Furthermore, by construction s is a square modulo n. Therefore (e, f, s) is the principal tweaked square root of h. The only remaining task is to show that the distribution of h is uniform.

Which choices of (e', f', x) lead to h? Write (e, f, s) for the principal tweaked square root of h. If $\gcd\{h, n\} = 1$ then $\gcd\{s, n\} = 1$ so $g = \gcd\{x, n\} = 1$; thus $e' = e$, $f' = f$, and x is one of the four square roots of s modulo n. If $\gcd\{h, n\} = p$ then $\gcd\{s, n\} = p$ so $g = \gcd\{x, n\} = p$; thus $e' = e$, $f' \in \{1, 2\}$, and x is one of the two square roots of s modulo n. If $\gcd\{h, n\} = q$ then $\gcd\{s, n\} = q$ so $g = \gcd\{x, n\} = q$; thus $e' \in \{-1, 1\}$, $f' = f$, and x is one of the two square roots of s modulo n. If $\gcd\{h, n\} = n$ then $\gcd\{s, n\} = n$ so $g = \gcd\{x, n\} = n$; thus $e' \in \{-1, 1\}$, $f \in \{1, 2\}$, and $x = 0$. To summarize, each integer $h \in \{0, 1, \ldots, n - 1\}$ is produced by *at most* four choices of (e', f', x). There are n possibilities for h and $4n$ possibilities for (e', f', x), so each integer $h \in \{0, 1, \ldots, n - 1\}$ is produced by *exactly* four choices of (e', f', x). In particular, each integer $h \in \left\{1, 2, \ldots, 2^K\right\}$ is produced by exactly four choices of (e', f', x).

|Principal|: h is uniform exactly as above, and (e, f, s) is the principal tweaked square root of h, so $(e, f, \min\{s, n - s\})$ is the |principal| tweaked square root of h. □

5 Generic Selective Inversion Using Many Signatures

Suppose we're given a public key pq, integers $h_1, h_2, \ldots, h_Q, h' \in \left\{1, 2, \ldots, 2^K\right\}$, and a tweaked square root of each h_i modulo pq. How quickly can we compute a tweaked square root of h' modulo pq? One approach is to factor pq; are there better approaches?

More formally: Fix $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Fix K and D. Fix $Q \geq 0$. For each algorithm A define $\text{PrInvSelective}_Q(A)$ as the chance that $A(pq, h_1, e_1, f_1, s_1, \ldots, h_Q, e_Q, f_Q, s_Q, h')$ is some $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq - 1\}$ satisfying $e' f' (s')^2 \equiv h' \pmod{pq}$, when

- (p, q) is a D-distributed random secret key,
- each h_i is a uniform random element of $\left\{1, 2, \ldots, 2^K\right\}$,
- (e_i, f_i, s_i) is an α-distributed random tweaked square root of h_i mod pq,
- h' is a uniform random element of $\left\{1, 2, \ldots, 2^K\right\}$,

and all of these choices are independent. How large can $\text{PrInvSelective}_Q(A)$ be, as a function of the resources consumed by A?

The answer is that this problem is provably as difficult as factorization of public keys. The construction of SimSign_Q is an easy generalization of last section's construction of SimSign_1. For example, here is $\text{SimSign}_Q(A)$ for $\alpha =$ unstructured:

0. Input n and h'.
1. For each $i \in \{1, 2, \ldots, Q\}$:
2. Generate a uniform random vector (e_i, f_i, s_i) in the usual range.

3. Compute $h_i = e_i f_i s_i^2 \bmod n$.
4. Go back to step 2 if $h_i \notin \{1, 2, \ldots, 2^K\}$.
5. Print $A(n, h_1, e_1, f_1, s_1, \ldots, h_Q, e_Q, f_Q, s_Q, h')$.

The remaining constructions work similarly.

Theorem 5.1. $\mathrm{PrInvBlind}(\mathrm{SimSign}_Q(A)) = \mathrm{PrInvSelective}_Q(A)$.

Proof. Exactly as in Section 4. □

6 Generic Existential Inversion: The Unstructured $B = 0$ Case

Suppose we're given a public key pq and integers $h_1, \ldots, h_{Q+1} \in \{1, 2, \ldots, 2^K\}$. We're allowed to adaptively select Q distinct i's and see tweaked square roots of the corresponding h_i's. Our goal is to compute a tweaked square root of the *other* h_i. How quickly can we do this?

More formally: Fix $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Fix K and D. Fix $Q \geq 0$. For each algorithm A define $\mathrm{PrInvExistential}_Q(A)$ as follows. A is given pq where (p, q) is a D-distributed random secret key, and uniform random elements $h_1, h_2, \ldots, h_{Q+1}$ of $\{1, 2, \ldots, 2^K\}$, all of these choices being independent. A makes Q distinct oracle queries i; in response to each i, A is given an α-distributed random tweaked square root (e_i, f_i, s_i) of h_i modulo pq, again independently of other choices. Now $\mathrm{PrInvExistential}_Q(A)$ is the probability that A outputs some $(i, e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq - 1\}$ such that $e' f' (s')^2 \equiv h_i \pmod{pq}$ and such that i was not one of the oracle queries.

The big difference between this generic-existential-inversion problem and the generic-selective-inversion problem in Section 5 is that we're now allowed to decide which of the h_i's will be easiest to attack. Does this make the problem easier? Perhaps we gain from the extra flexibility.

This section uses a new idea to show that there is no gain in the case of unstructured signatures. The reader might guess, after previous sections, that the proof constructs an algorithm for generic selective inversion or generic blind inversion; in fact, the proof jumps directly to the factorization problem. I don't know any way to get from a generic-existential-inversion algorithm to a generic-blind-inversion algorithm, in the case $B = 0$, except via factorization.

The new idea. Let's start by reviewing the standard proof that the gain is at most a factor $Q + 1$. Given a generic-existential-inversion algorithm A, build a generic-selective-inversion algorithm $\mathrm{GuessSelection}(A)$ that handles inputs $(n, h_1, e_1, f_1, s_1, \ldots, h_Q, e_Q, f_Q, s_Q, h')$ as follows:

- Choose a uniform random integer $\pi \in \{1, \ldots, Q + 1\}$.
- Insert h' at position π in the list $h_1, \ldots, h_Q$, and relabel the resulting list as $h_1, \ldots, h_{Q+1}$. Also relabel e_i, f_i, s_i accordingly.
- Run $A(n, h_1, \ldots, h_{Q+1})$, using e_i, f_i, s_i to answer query i from A; abort if A selects $i = \pi$ for a query rather than for output.

The choice of π is independent of the operation of A before an abort occurs, so this algorithm GuessSelection(A) aborts with probability exactly $Q/(Q+1)$. If GuessSelection(A) does not abort then it runs A with exactly the right input distribution.

This construction is the heart of the 1993 Bellare–Rogaway loose security proof. The random choice of π in GuessSelection(A) is a guess for the index i that A will use for its output; when a correct guess does occur, it makes the generic-existential-inversion problem equivalent to the generic-selective-inversion problem, eliminating the extra flexibility of the generic-existential-inversion problem.

Now let's feed this generic-selective-inversion algorithm GuessSelection(A) to the reductions in previous sections. Section 5 produces a generic-blind-forgery algorithm SimSign(GuessSelection(A)): each input h_i is replaced by an output from the appropriate simulator. Section 3 then produces a factorization algorithm RandSquare(SimSign(GuessSelection(A))): the input h' is replaced by a random efs^2, so that a tweaked square root of h' reveals a factorization of pq.

Wait a minute! What's happening to h_i is almost the same as what's happening to h'. In fact, with the unstructured simulator, what's happening to h_i is *exactly* the same as what's happening to h'! Why did we bother to distinguish h_i from h' in the first place? The new idea is to exploit unstructured signatures by treating all of the inputs $h_1, \ldots, h_{Q+1}$ the same way, directly producing a factorization algorithm; there is no need to guess which one is h', and there is no need for a detour through GuessSelection(A).

Here is the new, direct, almost ludicrously simple construction of a factorization algorithm RandSquares(A) from a generic-existential-inversion algorithm A for $\alpha =$ unstructured:

0. Input n.
1. For each $i \in \{1, 2, \ldots, Q+1\}$:
2. Generate a uniform random vector (e_i, f_i, s_i) in the usual range.
3. Compute $h_i = e_i f_i s_i^2 \bmod n$.
4. Go back to step 2 if $h_i \notin \{1, 2, \ldots, 2^K\}$.
5. Compute $(j, e', f', s') = A(n, h_1, \ldots, h_{Q+1})$, using (e_i, f_i, s_i) to answer query i from A. There is no possibility of aborting here; we have an answer for every i!
6. If $\gcd\{n, s' - s_j\} \notin \{1, n\}$, print it and stop.
7. If $\gcd\{n, s'\} \notin \{1, n\}$, print it and stop.

The time for RandSquares(A) is the time for A plus the time for the final gcd computations and, on average, the time for $(Q+1)n/2^K < 2(Q+1)$ generations of h_i.

Theorem 6.1. PrFactor(RandSquares(A)) $\geq (1/2)$ PrInvExistential(A) *if* $\alpha =$ unstructured.

Proof. Let (p, q) be a D-distributed random secret key. By construction the quantities $h_1, \ldots, h_{Q+1}$ inside (RandSquares(A))(pq) are independent uniform random elements of $\{1, 2, \ldots, 2^K\}$, so the event $e'f'(s')^2 \equiv h_j \pmod{pq}$ occurs with probability exactly PrInvExistential(A). Given this event, one of $s', s' - s_j$

has a nontrivial factor in common with pq with conditional probability at least 1/2, exactly as in Theorem 3.1. □

7 Generic Existential Inversion: The $B \geq 1$ Case

Fix $B \geq 0$. Suppose we're given a public key pq and (via an oracle) random access to $h_1(0), \ldots, h_1(2^B - 1), h_2(0), \ldots, h_2(2^B - 1), \ldots, h_{Q+1}(0), \ldots, h_{Q+1}(2^B - 1)$. We're allowed to adaptively select Q distinct i's; for each selected i we see a uniform random $r_i \in \{0, 1, \ldots, 2^B - 1\}$ and a tweaked square root of $h_i(r_i)$. Our goal is to compute some r and some tweaked square root of $h_i(r)$ for the remaining i. How quickly can we do this?

As usual, the answer depends on the tweaked-square-root distribution $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Section 6 discussed $\alpha =$ unstructured, and gave a tight security proof for unstructured signers for $B = 0$; this proof generalizes immediately to a tight security proof for unstructured signers for all B. The initial computations of $h_i(r)$ might sound overly time-consuming when B is large, because there are $2^B(Q+1)$ pairs (i, r); but these computations can be deferred until they are actually needed.

What about $\alpha \in \{\text{principal}, |\text{principal}|\}$? There is a tight security proof for all $B \geq 1$, coming from a different way to build a factorization algorithm MixedSim(A) out of a generic-existential-inversion algorithm A. This algorithm MixedSim(A), given n,

- chooses a uniform random r_i for each $i \in \{1, 2, \ldots, Q+1\}$;
- uses the α simulator to build $e_i(r_i), f_i(r_i), s_i(r_i), h_i(r_i)$;
- uses the *unstructured* simulator to build $e_i(r), f_i(r), s_i(r), h_i(r)$ for $r \neq r_i$;
- runs A, answering query i with $r_i, e_i(r_i), f_i(r_i), s_i(r_i)$;
- aborts if the output j, r', e', f', s' has $r' = r_j$; and
- tries $\gcd\{s', n\}$ and $\gcd\{s' - s_j(r'), n\}$ as factors of n.

This algorithm aborts with probability exactly $1/2^B$: r_j is independent of everything seen by A and therefore independent of r'. If the algorithm does not abort then it has conditional probability at least 1/2 of factoring n, exactly as in Theorem 3.1.

How powerful are claw-free permutation pairs? Readers should recognize the central idea of this construction—choosing a random r_i, building $h_i(r_i)$ according to the target simulator, and building $h_i(r)$ for $r \neq r_i$ to solve the underlying hard problem—as exactly the Katz–Wang idea used to prove [15, Section 4.1, Theorem 2].

The Katz–Wang theorem is stated for all "claw-free permutation pairs," following [14] and a suggestion of Dodis and Reyzin. One could directly apply the Katz–Wang theorem to the exponent-2 claw-free permutation pair defined by Goldwasser, Micali, and Rivest in [14, Section 6.3], obtaining a tight security proof for an alternate system that at first glance appears quite similar to Rabin–Williams. Unfortunately, a closer look shows that verification in the alternate system is even slower than verification of exponent-3 RSA signatures.

This alternate signature system is therefore of much less practical interest than the Rabin–Williams system.

Specifically, [14, Section 6.3] considers the permutations $x \mapsto \left|x^2 \bmod pq\right|$ and $x \mapsto \left|4x^2 \bmod pq\right|$ of the set of positive integers having Jacobi symbol 1 modulo pq. (Absolute values and "positive" here refer to integers between 1 and $(pq-1)/2$.) One can hash to this set by luck (if B is not very small), as Rabin did, or by tweaks, as Williams did. The verifier then has to check that s is a preimage of $H(m)$ under the first permutation: i.e., that s is positive, that s has Jacobi symbol 1, and that $\left|s^2 \bmod pq\right| = H(m)$. Unfortunately, the Jacobi-symbol computation takes much more time than squaring modulo pq.

Dropping the Jacobi-symbol requirement—in other words, switching back to Rabin–Williams signatures—speeds up verification but moves outside the world of permutation pairs; the wider range of accepted inputs means that the verifier's squaring map is no longer a permutation. One can recognize claws in the Rabin–Williams context, but they are claws between a 4-to-1 map and a 1-to-1 map, with two different algorithms for generating the inputs to the two maps. This is exactly where my simulators involved extra work.

Can the same idea be pushed to 0 bits of hash randomization? For $B = 0$, the MixedSim construction accomplishes nothing. It never uses the unstructured simulator; it always aborts. The construction needs at least one bit of hash-input randomization to separate the target simulator from the unstructured simulator. Eliminating the abort does not produce a security proof: if s_j was produced by (e.g.) the principal simulator then it is *not* a uniform random square root of its square and there is no reason to believe that $s' - s_j$ will have a factor in common with n.

However, for $\alpha =$ unstructured, eliminating the abort *does* produce a security proof, and further eliminating the selection of r_i produces exactly the new construction of Section 6. This is another way to see both the limitations and the power of the new idea in Section 6: the construction refuses to distinguish the α simulator from the unstructured simulator, and therefore requires $\alpha =$ unstructured, but the construction also skips the selection of r_i, and therefore can handle $B = 0$.

Tight security for principal $B = 0$ Rabin–Williams remains an open question. Switching from unstructured $B = 0$ signers to principal $B = 0$ signers breaks all of my tight security reductions, and presumably breaks any tight black-box reduction. A tight black-box reduction for principal $B = 0$ Rabin–Williams was claimed in [20, Section 6, Theorem 1], but [20, Section 6, Theorem 1, Proof, equality between "Pr(F successes)" and "$\epsilon(k)$"] implicitly assumes that attackers cannot distinguish principal square roots from arbitrary integers modulo pq.

8 Generic Attacks

Let's review three typical examples of attacks on the Rabin–Williams system:

- NFS factorization: The attacker uses the number-field sieve to factor pq into (p, q). The attacker then chooses a message m', chooses a B-bit string r',

computes $h' = H(r', m')$ using an oracle for H, uses (p, q) to compute a tweaked square root (e', f', s') of h', and forges the signature (e', f', r', s') of m'. This attack always succeeds, for all functions H. Fortunately, this attack is very slow when K is large.
- Signing leaks: The attacker chooses a message m and asks the signer for two signatures of m. The signer responds with (e_1, f_1, r_1, s_1) and (e_2, f_2, r_2, s_2). The attacker computes $\gcd\{s_2, n\}$ and $\gcd\{s_1 - s_2, n\}$, hoping to factor n and proceed as in the previous attack. In the case of variable unstructured $B = 0$ signers, this attack succeeds with probability $\geq 1/2$, for all functions H: notice that $r_1 = r_2$ since $B = 0$, and therefore that $e_1 f_1 s_1^2 \equiv e_2 f_2 s_2^2$; continue as in Theorem 3.1. Fortunately, this attack does not work for fixed signers, or for principal or |principal| signers, or for signatures with large B.
- MD5 collisions: The attacker finds distinct messages m, m' with $\text{MD5}(m) = \text{MD5}(m')$. The attacker asks the signer for a signature of m and then forges the same signature of m'. This attack works if $B = 0$ and H is determined by MD5, a surprisingly common situation in practice. Fortunately, one can easily change H to stop the attack.

Consider the class of "H-generic attacks" that work for all (or a significant fraction of all) functions H, given oracle access to H. This class includes many of the attacks in the literature, although there are also many exceptions; it does not include the MD5-collisions attack, for example, but it does include the factorization attack and the signing-leak attack.

How powerful are H-generic attacks against the Rabin–Williams system? Can they be better than factorization? Define PrAttack(A) as the average, over all functions H, of the success probability of A using an oracle for H. Can PrAttack(A) be much larger than the other probabilities considered in this paper, as a function of the resources consumed by A?

The signing-leak example shows that these attacks can be quite successful: variable unstructured $B = 0$ signers are broken by an extremely fast generic attack. But the picture is different for fixed signers. For fixed signers, generic attacks that see hash values of $Q+1$ distinct messages are as difficult as Q-query generic existential inversion. Given a generic-attack algorithm A, build a generic-existential-inversion algorithm FixSignatures(A) as follows: FixSignatures(A) runs A, keeps track of the distinct messages $m_1, m_2, \ldots, m_{Q+1}$ that are hashed, answers a hash query for (r, m_i) as $h_i(r)$, and answers a signature query for m_i by feeding i to its tweaked-square-root oracle. The distribution of signatures in this algorithm is identical to the distribution of signatures produced by a legitimate *fixed* signer, so FixSignatures(A)'s chance of success is the same as A's chance of success against fixed signers.

This FixSignatures construction is weaved through the Katz–Wang reduction "generic attack for fixed $B = 1$ RSA $\Longrightarrow$ blind RSA inversion," and can similarly be weaved through separate proofs of "generic attack for fixed unstructured $B \geq 0$ Rabin–Williams $\Longrightarrow$ factorization" and "generic attack for fixed principal

$B \geq 1$ Rabin–Williams $\Longrightarrow$ factorization" and so on; but this means repeating the same construction as part of every reduction. I learned the general principle "generic attack for fixed $\Longrightarrow$ generic existential inversion" from the illustrative "RSA1" and "RSA2" examples given by Koblitz and Menezes in [17, Sections 3.2 and 3.4].

In particular, generic attacks against fixed signers are as difficult as factorization whenever generic existential inversion is as difficult as factorization. Note also that variable signers are indistinguishable from fixed signers *if* B is large. The bottom line is that there cannot be any generic attacks better than factorization against fixed unstructured $B \geq 0$ Rabin–Williams, or against fixed principal $B \geq 1$ Rabin–Williams, or against fixed |principal| $B \geq 1$ Rabin–Williams, or against variable large-B Rabin–Williams.

Acknowledgments. Thanks to Dan Boneh for pointing out [15] to me shortly after it was posted. Thanks to Tanja Lange and the anonymous referees for many suggestions regarding exposition.

References

1. Ashby, V. (ed.): First ACM conference on computer and communications security, Association for Computing Machinery, New York (1993); See [5]
2. Atluri, V., Jaeger, T.: Proceedings of the 10th ACM conference on Computer and communications security. ACM Press, New York (2003); See [15]
3. Barua, R., Lange, T. (eds.): INDOCRYPT 2006. LNCS, vol. 4329. Springer, Heidelberg (2006); See [18]
4. Bellare, M. (ed.): CRYPTO 2000. LNCS, vol. 1880. Springer, Heidelberg (2000); See [10]
5. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for de- signing efficient protocols. In [1], 62–73 (1993); Citations in this document: §1, §1, §1
6. Bellare, M., Rogaway, P.: The exact security of digital signatures: how to sign with RSA and Rabin. In [21], 399–416 (1996); see also newer version [7]
7. Mihir Bellare, Phillip Rogaway, The exact security of digital signatures: how to sign with RSA and Rabin (1996) see also older version [6], `http://www-cse.ucsd.edu/~mihir/papers/exactsigs.html` Citations in this document: §2
8. Bernstein, D.J.: RSA signatures and Rabin–Williams signatures: the state of the art (2008), `http://cr.yp.to/papers.html#rwsota` Citations in this document:
9. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In [12], 30–45 (1997); Citations in this document
10. Coron, J.-S.: On the exact security of Full Domain Hash. In [4], 229–235. MR 2002e:94109. (2000), `http://www.eleves.ens.fr/home/coron/publications/publications.html` Citations in this document: §1
11. Coron, J.-S.: Optimal security proofs for PSS and other signature schemes. In [16], 272–287. (2002), `http://www.eleves.ens.fr/home/coron/publications/publications.html` Citations in this document: §1
12. Darnell, M.J. (ed.): Cryptography and Coding 1997. LNCS, vol. 1355. Springer, Heidelberg (1997); See [9]

13. Goh, E.-J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. Journal of Cryp- tology 20, 493–514 (2007); See [15]
14. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17, 281–308 (1988), http://theory.lcs.mit.edu/~rivest/publications.html Citations in this document: §1, §7, §7, §7
15. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In [2], 155–164 (2003) portions incorporated into [13], http://www.cs.umd.edu/~jkatz/papers.html Citations in this document: §1, §8, §8
16. Knudsen, L. (ed.): EUROCRYPT 2002. LNCS, vol. 2332. Springer, Heidelberg (2002); See [11]
17. Koblitz, N., Menezes, A.J.: Another look at "provable security" (revised May 4, 2005); see also newer version [19], http://eprint.iacr.org/2004/152/ Citations in this document: §1, §1, §8
18. Koblitz, N., Menezes, A.J.: Another look at "provable security". II. In [3], 148–175 (2006), http://eprint.iacr.org/2006/229 Citations in this document: §1
19. Koblitz, N., Menezes, A.J.: Another look at "provable security". Journal of Cryptology 20, 3–37 (2007); see also older version [17]
20. Kurosawa, K., Ogata, W.: Efficient Rabin-type digital signature scheme. Designs, Codes and Cryptography 16, 53–64 (1999); Citations in this document: §7, §7
21. Maurer, U.M. (ed.): EUROCRYPT 1996. LNCS, vol. 1070. Springer, Heidelberg (1996); See [6]
22. Stinson, D.R.: Some observations on the theory of cryptographic hash functions (2001), http://eprint.iacr.org/2001/020 Citations in this document: §1
23. Stinson, D.R.: A polemic on notions of cryptographic security (2004), http://www.cacr.math.uwaterloo.ca/~dstinson/pubs.html Citations in this document: §1
24. Yung, M. (ed.): CRYPTO 2002. LNCS, vol. 2442. Springer, Heidelberg (2002)

Threshold RSA for Dynamic and Ad-Hoc Groups*

Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin

IBM T.J.Watson Research Center
Hawthorne, NY USA

Abstract. We consider the use of threshold signatures in ad-hoc and dynamic groups such as MANETs ("mobile ad-hoc networks"). While the known threshold RSA signature schemes have several properties that make them good candidates for deployment in these scenarios, none of these schemes seems practical enough for realistic use in these highly-constrained environments. In particular, this is the case of the most efficient of these threshold RSA schemes, namely, the one due to Shoup. Our contribution is in presenting variants of Shoup's protocol that overcome the limitations that make the original protocol unsuitable for dynamic groups. The resultant schemes provide the efficiency and flexibility needed in ad-hoc groups, and add the capability of incorporating new members (share-holders) to the group of potential signers without relying on central authorities. Namely, any threshold of existing members can cooperate to add a new member. The schemes are efficient, fully non-interactive and do not assume broadcast.

1 Introduction

A distributed signature scheme is a protocol where the ability to sign is distributed among a group of entities, so that a sufficiently large subset can produce valid signatures while a "small" subset cannot generate such a signature. These schemes are often referred to as t [illegible] n [illegible] where n is the total number of entities and t is the "threshold". Namely, $t+1$ cooperating parties can produce a valid signature, but t or less cannot (even if they depart maliciously from the protocol). Threshold signature schemes are known for standard signatures such as RSA and DSS. One major appeal of these schemes is that the verification of a signature uses a regular public key and a standard verification procedure; hence the verifier of a signature does not need to be aware of the form (centralized or distributed) in which the signature was generated, or who were the parties involved, nor does the signature increase in size as a function of the number of signers.

Typically, in these systems each signing entity holds a share of the signing key, that it uses to produce a "fragment" of a signature on a given message.

* Full version available in [15].

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 88–107, 2008.

When a sufficient number of such fragments are collected (i.e., a threshold), the fragments are combined in some prescribed manner to generate the resultant (standard) signature on the given message.

Threshold signature were traditionally motivated by applications requiring the protection of highly-valuable signature keys, such as in the cases of a certification authority signing public-key certificates or a bank minting electronic coins. In such applications, threshold signatures increase the security of the key by preventing "a single point of failure", and also increase the reliability and availability of the service (since disabling some nodes does not disrupt the service as long as there are $t + 1$ good functioning nodes). A substantial body of work has been devoted to designing efficient threshold signature schemes [3,5,9], especially for standard algorithms such as RSA [11,8,19,14,13,24] and DSS [6,17]. Given this motivation, prior works were mostly concerned with scenarios with a small number of nodes, with static configuration and tight coordination. (Typically the protocol would be implemented by a small set of nodes, all of which are governed by one administrative entity.)

A more recent application of threshold signatures has emerged in the area of networking and distributed computing, such as in the setting of mobile and ad-hoc networks (MANETs). In these cases, relatively small subsets of very large (and dynamic) groups report data, that is aggregated and "certified" by means of a signature. Examples include vehicular networks where cars report traffic conditions, sensor networks that report aggregate data such as temperature or radiation levels, military devices transmitting information to be reported to various commands, and more. Threshold signatures provide a robust, flexible and secure way for the nodes to report and certify data that can be verified by third parties regardless of the specific reporters. All that is needed is the assurance that a large enough subset of authorized reporters agreed on the data.

In these environments the set of parties is formed in dynamic and ad-hoc manners. Nodes may be dynamically added to the system, their share of the secret key can be either installed by trusted authorities before deployment, or added "on-the-fly" by a qualified subset of nodes already in the network. Large numbers of nodes may be deployed in the network, and yet at a given time a node may be in the communication range of only a few other nodes. In many such applications, communication bandwidth may be constrained (e.g., due to energy limitations), transmitting large amount of data or heavy interaction may be infeasible, and expensive communication primitives like broadcast may not be available. Adapting threshold signature schemes to work in such environments is challenging. In principle, threshold RSA signatures [11,8,24] are appealing in this case due to two important properties:

Standard signatures. They implement standard RSA signatures. Namely, the end-result of running these protocols is a standard RSA signature on the given message, and anyone can verify that signature as if it was generated by a standard centralized signer.

Non interactive. Given a message and its share of the secret key, each party locally computes a "signature fragment" without any interaction with the other parties. Then there is a public combination function that takes all these signature fragments (together with the message and public key) and turns them into a standard RSA signature.

Note that we would like the scheme to remain non-interactive even when misbehaving parties provide wrong signature fragments. For these cases, techniques from [19,18] can be used to obtain fragment verifiability without losing the non-interactive nature of the protocols.

From the existing protocols with the above properties, the most practical is the one due to Shoup [24], which is very efficient in scenarios with static, relatively small sets of parties. But Shoup's protocol has a parameter n, which is a global upper-bound on the number of (potential) parties in the protocol, and the computation of signature fragments and the combining function use the number $n!$ in the calculations (specifically as an exponent in a modular exponentiation operation). This means that the parameter n must be fixed and known to all parties, and the computation takes time at least linear in n. As the parameter n grows, as is likely in the dynamic applications that we mentioned, these computations become expensive or even infeasible.

This problem is even more serious, since in Shoup's protocol n is an upper bound on the [illegible] of parties in the protocol, namely it is assumed that no party has an identity whose value (as an integer) is larger than n. For example, if the identities are arbitrary 32-bit numbers, the protocol must use $n = 2^{32}$ so $n!$ is a 2^{37}-bit number! (Clearly, using network addresses or serial numbers of 64 bits or 160-bit hash values is incompatible with this protocol.) The range of identities can be reduced via tight coordination of the identity name space, but such tight coordination flies in the face of the flexibility that is expected in dynamic groups. Further, tight coordination may be impossible in applications that need flexible addition of nodes, either by loosely-coordinated "trusted authorities" or even by completely un-coordinated groups of members within the group itself.

[illegible] We describe two results that extend Shoup's protocol to dynamic ad-hoc groups:

- We present a variant of Shoup's protocol that keeps all the appealing properties of the original scheme but frees the protocol from any dependency on the total number of parties. Technically, the dependency on n as discussed before is replaced with a dependence on t (where $t + 1$ is the number of parties that needed to generate a signature).
- We show the practicality of our scheme in the dynamic group scenarios by extending our threshold scheme to support the addition of new members without the need to centrally coordinate this join operation. Basically, we allow any set of $t+1$ or more parties to cooperate (non-interactively!) in order to add a new member into the group, without having to invoke any "trusted authority". This is done by adapting to our case the elegant non-interactive solution of Saxena et al. [25].

We believe that the combination of both results leads to the first practical solution for dynamic and communication-constrained scenarios described above.

2 Background

In this work we use RSA moduli of special form, namely, the product of [illegible] [illegible]: $N = pq$ such that $p = 2p' + 1$ and $q = 2q' + 1$ with p', q' also primes. We assume that $|p| = |q| = k$, where k is the security parameter. With Z_N^* we denote the set of non-negative integers smaller than N which are relatively prime to N. This is a group with respect to multiplication mod N. Define $\lambda(N) \stackrel{\text{def}}{=} 2p'q'$ (the Carmichael function of N). We also denote $m \stackrel{\text{def}}{=} p'q'$.

The RSA function [21] is defined by an integer e which is relatively prime to $\lambda(N)$:

$$\forall x \in Z_N^* \quad RSA_{N,e}(x) = x^e \bmod N$$

Since e is relatively prime to $\lambda(N)$ this is a permutation over Z_N^*. It is believed that for a randomly-generated composite N (which is a product of two large enough safe primes), inverting the RSA function on random inputs is infeasible. On the other hand, inverting this function is easy given some trapdoor information. The trapdoor could be the prime factorization of N, since it would allow to compute $d = e^{-1} \bmod \lambda(N)$ and hence

$$\text{if } y = x^e \bmod N \text{ then } x = y^d \bmod N$$

Assumption 1 (RSA). [illegible] Gen [illegible] (N, e) [illegible] (T, ϵ) [illegible] $\mathcal{A}$ [illegible] T[illegible]

$$\Pr[(N, e) \leftarrow \mathsf{Gen},\ x \in_R Z_N^*\ ;\ \mathcal{A}(N, e, y = x^e \bmod N) = x] \leq \epsilon$$

[illegible] We recall the [illegible] (FDH-RSA) signatures. The public key is (N, e), the secret key is $d = e^{-1} \bmod \lambda(N)$, and a hash function H which maps arbitrary messages to Z_N^* is also part of the public key. To sign a message M, the signer computes $y = H(M)$ and the signature $\sigma = y^d \bmod N$. To verify a message/signature pair (M, σ) under public key (N, e), the receiver checks if $\sigma^e = H(M) \bmod N$. It is well known that the security of FDH-RSA can be reduced to the RSA Assumption if we model H as a random oracle [4,7].

2.1 Threshold Cryptography

In a [illegible] scheme, the secret key of a cryptographic scheme is stored in a shared form among several parties [3,5,10]. The goal is to prevent compromise of the secret key by an attacker who breaks into parties and reads their memory. Indeed if the secret key were stored in a single party, then a single break-in would compromise the security of the entire scheme. The idea of threshold cryptography is to share the key among several parties so that the attacker must break into several of them before learning the secret key.

Threshold cryptography schemes are thus composed of two phases. The first is a [illegible] in which the secret key is installed in this shared form among the parties; this part can be performed by a trusted dealer who temporarily knows the secret key, and shares it among the parties, before erasing it from its memory. (Alternatively the parties jointly generate the key directly in a shared form.) The second phase is a [illegible], where the parties run some protocol to jointly perform the cryptographic computations in which the secret key is employed.

We assume to have a set of upto n [illegible] that can communicate with each other. We do not assume physically secure channels or a broadcast channel among them. In addition to the n parties, we also consider a [illegible] who will share the secret key among them (we do assume that for this phase there are private channels between the dealer and the parties).

We assume a computationally bounded adversary, $\mathcal{A}$, who can corrupt up to t of the n parties in the network, where t is a parameter. (In our application domain we typically have $t \ll n$, but the protocol that we describe works for any $t < n$.) We call an adversary that corrupts no more than t parties a t-adversary. We say that the adversary is [illegible] if it corrupts all its parties at the beginning of the protocol, otherwise we say it is [illegible]. An adversary is [illegible] if it does not modify the code of the corrupted parties, but just reads their memory. A [illegible] adversary on the other hand may also cause corrupted parties to behave in any (possibly malicious) way. Yet, the adversary can never corrupt the dealer.

Threshold Signature Schemes. The following definitions of secure threshold signature schemes are essentially taken from [17]. A threshold signature scheme is a pair of protocols (Thresh-Key-Gen, Thresh-Sig) for a set of n parties and a dealer, and a verification algorithm Ver.

Thresh-Key-Gen is a key generation protocol carried out by a designated dealer to generate a pair (pk, sk) of public/private keys. At the end of the protocol the private output of party P_i is a value sk_i (related to the private key sk). The public output of the protocol contains the public key pk, and possibly some additional verification information v.

Thresh-Sig is the distributed signature protocol. The private input of P_i is the value sk_i. The public inputs for all parties consist of a message M, the public key pk, and the verification information v (if any) . The output of the protocol is a signature sig on M (relative to the public key pk). The verification algorithm Ver, on input M, sig, pk, checks if sig is a valid signature of M under pk.

The definition of security is adapted from the centralized case. Specifically, we consider a t-adversary that can interact with the honest parties and ask them to generate signatures on messages $M_1, M2, \ldots$ that the adversary chooses adaptively. In the malicious model the adversary can also actively participate in these signature-generation protocols (via the set of t parties that it controls), and in either model the adversary can choose the set of parties that would participate in the current signature generation. As usual, the goal of the adversary is to produce a signature on any message M that was not obtained via one of these runs of the protocol.

Definition 1. [illegible] Thresh-Key-Gen Thresh-Sig Ver [illegible] t secure [illegible] (T, ϵ) [illegible] t [illegible] $\mathcal{A}$ [illegible] T [illegible] M [illegible] ϵ [illegible] $\mathcal{A}$ [illegible] $M_1, M_2, \ldots$ [illegible] $\mathcal{A}$ [illegible] $M \neq M_i$ [illegible] i [illegible]

2.2 Secret Sharing vs. Threshold Cryptography

Notice the difference between threshold cryptography and the simpler task of secret sharing [22,2]. In secret sharing, the secret is first shared among the parties who later reconstruct it. In threshold cryptography, the secret is never reconstructed, but rather employed as a shared input into a cryptographic computation. However the sharing mechanisms are usually similar in both areas.

Here we recall Shamir's scheme for secret sharing [22]. Let q be a prime and the secret is an integer $s \in Z_q$. The goal is to share s among n parties in such a way that t or less of them have no information about s, while $t+1$ of them can easily reconstruct it.

The dealer chooses t random values $a_t, \ldots, a_1$ in Z_q and considers the polynomial $f(x) = a_t x^t + \ldots + a_1 x + s$. Assume each party is given as a "name" an integer between 1 and n. Then party i is given the [illegible] $s_i = f(i) \bmod q$. Notice how t shares give no information about s (for every possible secret s' there is a polynomial f' consistent with s' and the t shares). On the other hand $t+1$ shares completely define s, via polynomial interpolation. Notice that if $s_{i_1}, \ldots, s_{i_{t+1}}$ are $t+1$ shares corresponding to the points in $S = \{i_1, \ldots, i_{t+1}\}$, then the secret can be computed from S and these shares as

$$s = f(0) = \sum_{j=1}^{t+1} L_S(0, i_j) s_{i_j} \bmod q$$

where $L_S(0, i_j)$ are the appropriate Lagrangian coefficients, namely

$$L_S(\alpha, \beta) \stackrel{\text{def}}{=} \frac{\prod_{\gamma \in S, \gamma \neq \beta} (\alpha - \gamma)}{\prod_{\gamma \in S, \gamma \neq \beta} (\beta - \gamma)} \bmod q \tag{1}$$

Notice that Shamir's secret sharing has an interesting homomorphic property that allows the shares to be used to jointly compute exponentiations of the form y^s where y is known, without reconstructing the secret in the clear. Assume for example that y is an element in a cyclic group G of known prime order q: then each party could publish the value $\gamma_i = y^{s_i}$ in G and then

$$y^s = y^{\left(\sum_{j=1}^{t+1} L_S(0, i_j) s_{i_j} \bmod q\right)} = \prod_{j=1}^{t+1} \gamma_{i_j}^{L_S(0, i_j)} \quad \in G$$

2.3 Shoup's Threshold RSA

The homomorphic property of Shamir's secret sharing described above would seem ideal for the setting of threshold RSA, as it would allow the parties to jointly compute RSA signatures with secret key d, without recovering d itself (recall that to compute a signature on M, the parties have to jointly compute $\sigma = y^d \bmod N$ where $y = H(M)$). However, a closer look reveals that this approach does not immediately work in the case of threshold RSA.

The problem is that the sharing must be conducted modulo $\lambda(N)$, which must be kept secret from the parties themselves (since knowing $\lambda(N)$ or a multiple of it would allow any single party to factor N and compute the secret key d for itself). This issue arises in the computation of the Lagrangian coefficients modulo $\lambda(N)$, since they require the computation of inverses modulo $\lambda(N)$ (which are equivalent to knowing a multiple of $\lambda(N)$). This problem received considerable attention in the threshold cryptography literature (e.g. [11,8]) and various solutions were proposed. Shoup in [24] presented the most efficient solution to date.

Shoup observed that if we denote $\Delta = n!$ then the values $\Delta \cdot L_S(0, j)$ for all S, j are integers and no inverse computation modulo $\lambda(N)$ is required if we compute the linear combination of the shares using $\Delta \cdot L_S(0, i_j)$ instead of $L_S(0, i_j)$. However doing so we will reconstruct the value $\sigma' = y^{\Delta d} \bmod N$ rather than $\sigma = y^d \bmod N$. But if e relatively prime to Δ, we can apply the extended Euclidean algorithm in the exponent [23] to recover σ from σ'. Details of Shoup's scheme follow.

Sharing Phase: Given the public key N, e, the dealer (who knows the factorization of N) computes $d = e^{-1} \bmod m$. It also chooses t random values $a_1, \ldots, a_t$ in Z_m and defines the polynomial $f(z) = a_t z^t + \ldots + a_1 z + d$. Party i is given the [illegible] $d_i = f(i) \bmod m$. The (maximal) number n of parties is fixed and public and with it the value $\Delta = n!$.

Signature Computation Phase: On input a message M, party i computes $y = H(M) \in Z_N^*$ and the [illegible] $\sigma_i = y^{2\Delta \cdot d_i} \bmod N$ which it then publishes. (See the next section for an explanation of the 2Δ factor in the exponent.) Then given any $t+1$ of these values, $\sigma_{i_1}, \ldots, \sigma_{i_{t+1}}$, anybody can compute

$$\sigma' = \prod_{j=1}^{t+1} \sigma_{i_j}^{2\Delta \cdot L_S(0,i_j)} \bmod N \qquad (2)$$

By simple algebra we have that

$$\sigma' = \prod_{j=1}^{t+1} y^{4\Delta^2 \cdot L_S(0,i_j) \cdot d_{i_j}} \bmod N =$$

$$(y^{4\Delta^2})^{\sum_{j=1}^{t+1} L_S(0,i_j) d_{i_j} \bmod m} \bmod N = y^{4\Delta^2 \cdot d} \bmod N$$

Notice that the operations (polynomial interpolation) "in the exponent" are performed mod m because the value $y^{4\Delta^2}$ has order m.

At this point we need to show how to compute $\sigma = y^{1/e}$, given σ'. If $GCD(e, 4\Delta^2) = 1$ (which we ensure by choosing e as a prime larger than n), we compute integers a, b such that $ae + 4b\Delta^2 = 1$ and set $\sigma = y^a(\sigma')^b \bmod N$. To see that $\sigma = y^{1/e} \bmod N$ note the following equalities mod N:

$$\sigma = y^a(\sigma')^b = (y^{1/e})^{ae} \cdot (y^{1/e})^{4b\Delta^2} = (y^{1/e})^{ae+4b\Delta^2} = y^{1/e}.$$

3 Threshold RSA Signatures for Ad-Hoc Groups

As discussed in the Introduction, we are interested in constructing a threshold RSA signature for "ad hoc" groups, that can be very large, dynamic, and decentralized, and where parties from very different domains aggregate temporarily to perform a specific ("ad hoc") task. In these networks, because of the large number of parties and the loose coordination among them, it is impractical (if not impossible) to maintain a coherent centralized naming scheme, i.e., one where if there are n parties in the network their identities will be integers from 1 to n.

Consider for example the task of adding parties to the network and giving them shares of the secret key. In Section 5 we describe a solution in which $t+1$ honest parties can provide a new party with a new share of the secret key. To do that, the $t+1$ parties must give this new party a unique identity ID and its share $f(ID)$, where f is the t degree polynomial such that $f(0) = d$. In a very decentralized network, it is basically impossible to keep track of the IDs issued by various subsets of parties in the network, and therefore it is infeasible to maintain these identities in a small subset of integers.

What is most likely to happen in ad hoc networks is that parties have an ID already assigned to them, probably a large integer (say a 32-bit or a 160-bit integer, e.g. their serial number, their IP address or a hash of some other identity) and we would want to use that ID all across the board. In the example above, when adding parties to the network, the subset of parties could use the ID of the new party (if it has one) or generate a random one for it (if the ID are sufficiently long, it will most likely be unique).

Now consider what happens in Shoup's scheme when the IDs of parties are long integers say between 1 and 2^k for some parameter k. In this case the computation of the value $\Delta = n!$ is infeasible as one must set $n = 2^k$ in order for Δ to remove all possible denominators in Eq. (2) (note that $2^k! > 2^{2^k}$). There are two places where this factor Δ is used in Shoup's scheme:

(1) In the interpolation of partial signatures: Given the various shares $\sigma_i = y^{2\Delta f(i)}$, the players reconstruct the value $\sigma' = y^{4\Delta^2 \cdot f(0)}$ via interpolation "in the exponent". One of the two Δ factors in the exponent is needed for the correct operation of the scheme, as it is exactly what lets the parties replace the fractional Lagrangian coefficients with integers that they can compute.

Reducing this Δ factor is straightforward. Indeed, given the set S of parties in the threshold computation of the RSA signature, this factor can be easily replaced by

$$\Delta_S \stackrel{\text{def}}{=} \text{lcm}\left\{ \left(\prod_{\substack{j \in S \\ j \neq i}} (i-j) \right) \ : \ i \in S \right\}.$$

Note that if S has $t+1$ elements, each a number between 0 and 2^k, then Δ_S is bounded by

$$\Delta_S \ < \ \prod_{\substack{i,j \in S \\ i \neq j}} (i-j) \ < \ (2^k)^{t^2} \ = \ 2^{kt^2},$$

so the bit-size of Δ_S is linear in k and at most quadratic in t.

(2) In the computation of partial signatures: Given public input y and their private share d_i, each party computes $\sigma_i = y^{2\Delta d_i} \bmod N$. This additional Δ factor in the exponent is introduced to obtain a provably secure scheme. Indeed we need to show that the values transmitted by the good parties do not help the adversary, beyond the computation of $\sigma = y^{1/e}$. This is done by a simulation argument in which we show that the adversary is able to compute the values transmitted by the good parties from any t shares (belonging to the parties the attacker controls) and the value $\sigma = y^{1/e}$. Assume that the adversary knows t shares $d_{i_1}, \ldots, d_{i_t}$, and denote $B = \{i_1, \ldots, i_t\}$ (where B stands for "bad") the set of corrupted parties. Denote $\tilde{B} = B \cup \{0\}$. Notice that any other share d_i for $i > t$ can be computed as

$$d_i = L_{\tilde{B}}(i,0)d + \sum_{j=1}^{t} L_{\tilde{B}}(i,i_j)d_{i_j} \bmod m$$

Thus the adversary given the value $\sigma = y^{1/e}$ can compute the value σ_i for $i > t$ output by a good party as follows:

$$\sigma_i = y^{2\Delta d_i} = (y^2)^{L_{\tilde{B}}(i,0)d + \sum_{j=1}^{t} L_{\tilde{B}}(i,i_j)d_{i_j} \bmod m} = \sigma^{2\Delta L_{\tilde{B}}(i,0)} \prod_{j=1}^{t} (y^{2d_j})^{\Delta L_{\tilde{B}}(i,j)}$$

Notice that the products $\Delta L_{\tilde{B}}(i,j)$ are all integer values and thus the adversary can perform this computation (since it does not require computing inverses modulo m). However, this argument would not hold if the parties transmitted just $\sigma_i = y^{d_i}$ (in which case we do not know if a simulation would be possible).

Eliminating this factor is not as easy as in case (1). The reason that we cannot replace Δ with some Δ_S as above is that the relevant set S here is the set $\tilde{B}$ of "bad parties" (i.e., those that were compromised by the adversary), and $\tilde{B}$ is not known to the honest parties. Fortunately, we show below that the analysis can still be carried out when Δ is replaced by the much smaller factor 2^{kt}. For this crucial reduction in the size of the exponent, we pay either by having to carry out the analysis in the random-oracle model or by strengthening the hardness assumption. (Note however that the basic Shoup scheme already relies on the random-oracle model for the non-interactive verification of signature fragments.)

Remark: Shoup presented an alternative protocol for threshold RSA in [24], where the share of party i is $d_i = \Delta^{-1} f(i) \bmod m$ (rather than $d_i = f(i)$).

Thereafter the parties do not have to raise their partial signatures to a factor Δ, as it is already "factored in" in their shares. However the dealer still needs to compute $\Delta^{-1} \bmod m$, and the signature generation still needs to exponentiate to the power of $\Delta \cdot L_S(\cdot,\cdot)$. Hence both the dealer and the "signature combiner" work in time at least linear in n (which is exponential in settings where $n = 2^k$).

3.1 Our Threshold RSA Protocol for Ad-Hoc Groups

We now describe the details of our new scheme, which we call **TFDH-RSA**. We first describe a basic version that is only secure in the honest-but-curious attack model.

Sharing Phase: Given the public key N, e (where e is a prime larger than $n = 2^k$) and the secret key d (such that $d = e^{-1} \pmod{m}$), the dealer chooses t random values $a_1, \ldots, a_t$ in Z_m and defines the polynomial $f(x) = a_t x^t + \ldots + a_1 x + d$. Each party i is given the share $d_i = f(i) \bmod m$.

Signature Computation Phase: On input a message M, party i computes $y = H(M) \in Z_N^*$ and the $\sigma_i = y^{2^{kt} d_i} \bmod N$, which it then publishes.

Given $t+1$ of these values $\sigma_{i_1}, \ldots, \sigma_{i_{t+1}}$ the signature $\sigma = y^d \bmod N$ is computed as follows. Let

$$\Delta_S = \mathrm{lcm}\left\{ \left(\prod_{\substack{j \in S \\ j \neq i}} (i - j) \right) \ : \ i \in S \right\} \tag{3}$$

where $S = \{i_1, \ldots, i_{t+1}\}$, and set

$$\sigma' = \prod_{j=1}^{t+1} \sigma_{i_j}^{\Delta_S \cdot L_S(0, i_j)} \bmod N. \tag{4}$$

Using extended Euclidean algorithm find values a, b such that $ae + b(2^{kt}\Delta_S) = 1$. Finally, the signature $\sigma = y^d \bmod N$ is computed as $\sigma = y^a \cdot (\sigma')^b \bmod N$.

Note that for any j, the quantity $\Delta_S \cdot L_S(0, i_j)$ is an integer that can be computed just by knowing S and i_j (no need to know m or any inverse $\bmod\ m$). Moreover, this integer is smaller than 2^{kt^2} and then the computation of σ' is feasible. To see that the value σ we computed is indeed $y^d \bmod N$ note that similarly to the case of Eq. (2)

$$\sigma' = y^{\Delta_S \cdot 2^{kt} \cdot f(0)} = y^{\Delta_S \cdot 2^{kt} \cdot d} \bmod N \tag{5}$$

and then

$$\sigma \ = \ y^a \cdot (\sigma')^b \ = \ (y^{1/e})^{ae} \cdot (y^{1/e})^{b 2^{kt} \Delta_s} \ = \ y^{1/e} \pmod{N}. \tag{6}$$

Finally, note that the values a, b computed via the extended Euclidean algorithm exist since by choice of e, $GCD(e, 2^{kt}\Delta_S) = 1$.

e The value e must be chosen as a prime larger than any possible identity (e.g., larger than 2^k). Alternatively, if the identities are random k-bit integers, the value e can be chosen somewhat smaller (but still a "large enough prime") and then rely on the fact that with high probability no two identities will have a difference that is divisible by e.

4 Security Analysis

Ideally, we would like to claim that the protocol above is as secure as the underlying signature scheme. In particular, we want to claim that seeing the signature fragments σ_i does not gives the adversary any more information than the signature σ itself, by presenting a simulator $\mathcal{S}$ that given σ can generate the view of the adversary in the protocol (i.e., the signature fragments of the honest parties $\sigma_j = x^{f(j)} \bmod N$).

Unfortunately, as we explained above, we do not know how to generate this view without the extra factor of Δ in the signature generation. Specifically, the simulator can only compute the related quantities $\sigma'_j = \sigma_j^{\Delta_{\tilde{B}}}$, where $\Delta_{\tilde{B}}$ is as defined in Eq. (3) (with respect to the set $\tilde{B}$ that consists of zero and the "bad parties"). The reason is that we can write

$$\sigma'_j \;=\; \sigma^{\Delta_{\tilde{B}} \cdot L_{\tilde{B}}(j,0)} \cdot \prod_{i \in B} y^{\Delta_{\tilde{B}} \cdot L_{\tilde{B}}(j,i) \cdot d_i},$$

and for all i, the quantity $\Delta_{\tilde{B}} \cdot L_{\tilde{B}}(j,i)$ is an integer that the simulator can efficiently compute from $\tilde{B}$, i and j. However, in this case we do not know a way to "extract" the $\Delta_{\tilde{B}}$ root out. In particular we do not know a "public exponent" corresponding to $f(j)$ that would let us use the GCD calculations in order to eliminate the extra factor of $\Delta_{\tilde{B}}$ in the exponent.

In dealing with this problem, we separate the treatment of the powers of two in $\Delta_{\tilde{B}}$ from the odd factors of it.[1] Write

$$\Delta_{\tilde{B}} = 2^{\ell(\tilde{B})} e(\tilde{B})$$

with $e(\tilde{B})$ odd. Clearly, for any set $\tilde{B}$ of $t+1$ elements in $[1..2^k]$, it must be that $\ell(\tilde{B}) < kt$.

Since in our protocol the parties compute signature fragments by raising $y = H(m)$ to the power $2^{tk} f(i)$, then the simulator does not need to take $2^{\ell(\tilde{B})}$ roots (see details below). On the other hand the problem of taking $e(\tilde{B})$ roots remains.

We present two solutions for this problem. In Theorem 2 we show that when we model the function H as a random oracle, our protocol can be proven t-secure (as per Definition 1) under the standard RSA assumption. This is a strong assurance of security, especially since random-oracle proofs are the only security assurance that is known for most standard RSA signatures. However, this proof does not say much about the security of our signature protocol when instantiated with

[1] See discussion after Theorem 1 about the reason for this separation.

any specific hash function. In particular, it still leaves open the possibility that there are hash functions H for which centralized RSA signature are secure but our protocol is not. Therefore, in Theorem 1 we provide a different analysis [illegible], relating the security of our protocol with any specific function H to a slightly modified centralized RSA signature that uses the same function H. We argue informally that this "slightly modified" scheme is likely to be as secure as the original one, hence providing yet other assurance of the security of our protocol.

4.1 Security Theorems

Consider the following signature scheme D-RSA: the public key is (N, e) where e is prime, and relatively prime with $\lambda(N)$. The secret key is the factorization of N. The public key contains a hash function H which outputs elements of Z_N^*. Moreover we allow the adversary to specify an additional parameter e' that must be an odd integer co-prime with e.

On input a message M, the signer returns a pair (σ, σ') such that $\sigma^e = (\sigma')^{e'} = H(M) \bmod N$. Given an alleged signature (σ, σ') on M, however, the verifier only checks that $\sigma^e = H(M) \bmod N$.

We say that the D-RSA signature scheme using the hash function H is [illegible] if no feasible forger $\mathcal{F}$ can win the following game: first $\mathcal{F}$ is given as input N, e and it specifies an odd integer e', that must be co-prime with e (Phase 1). Then $\mathcal{F}$ conducts a traditional adaptive chosen-message attack (i.e. $\mathcal{F}$ gets signatures on messages of its choice) and produces a valid signature on a message that it did not request before (Phase 2).

Theorem 1. [illegible] t [illegible] $\mathcal{A}$ [illegible] H [illegible] H [illegible] $\mathcal{A}$ [illegible] H [illegible]

[illegible] □ Although we cannot prove that the protocol TFDH-RSA does not give the adversary any more power than just interacting with the underlying RSA signature scheme (with the same hash function H), Theorem 1 tells us that it does not give it more power than what it could get from the ability to get also e'-th roots (in addition to the e-th roots that it gets from the signature scheme). It is generally believed that when e, e' are co-primes, then extracting e'-th roots does not help in extracting e-th roots. In this light, Theorem 1 can be interpreted as asserting that any attack on the protocol TFDH-RSA must either break the underlying RSA signatures, or find a way to use e'-th roots in order to extract e-th roots.

[illegible] Assume by contradiction that TFDH-RSA is not secure. Then there exists an adversary $\mathcal{A}$ that interacts with TFDH-RSA, statically corrupting at most t parties, such that $\mathcal{A}$ has a noticeable chance ϵ of forging an RSA signature. We want to use this $\mathcal{A}$ as a subroutine to construct a forger $\mathcal{F}$ that breaks the two-phase security of D-RSA. We want $\mathcal{F}$ to have a similar running time and similar success probability as $\mathcal{A}$.

The forger $\mathcal{F}$ basically simulates TFDH-RSA for the adversary $\mathcal{A}$. It is given a public key (N, e), and must run Phase 1. $\mathcal{F}$ starts by giving (N, e) to the adversary $\mathcal{A}$. The latter responds by asking to corrupt a set $B = \{i_1, \ldots, i_t\}$ of parties. ($\mathcal{A}$ can compromise upto t parties, and we assume w.l.o.g. that it compromises exactly t parties.) Then $\mathcal{F}$ chooses t values $d_{i_1}, \ldots, d_{i_t}$ at random in the interval $[N/4]$ and gives to $\mathcal{A}$ the value d_{i_j} as the secret-key share for party i_j. As observed above, the distributions of the shares given by $\mathcal{F}$ to $\mathcal{A}$, is statistically close to the distributions of the shares seen by $\mathcal{A}$ during a real execution of FDH-RSA.

With B the set of corrupted, parties, $\mathcal{F}$ sets $\tilde{B} \stackrel{\text{def}}{=} B \cup \{0\}$ and computes $\Delta_{\tilde{B}}$ as defined in Eq. (3), namely

$$\Delta_{\tilde{B}} = \operatorname{lcm}\left\{ \left(\prod_{\substack{j \in \tilde{B} \\ j \neq i}} (i - j) \right) : i \in \tilde{B} \right\}.$$

It also lets $\ell(\tilde{B})$ be the largest integer ℓ such that 2^ℓ divides $\Delta_{\tilde{B}}$, and sets $e(\tilde{B}) = \Delta_{\tilde{B}}/2^{\ell(\tilde{B})}$ and $\Gamma_{\tilde{B}} = 2^{kt} \cdot e(\tilde{B})$ (so $e(\tilde{B})$ is odd and $\Delta_{\tilde{B}}$ divides $\Gamma_{\tilde{B}}$). The forger $\mathcal{F}$ concludes Phase 1 by specifying the value $e' = e(\tilde{B})$. (Note that e' is co-prime with e, since e is a prime larger than 2^k and all the identities in $\tilde{B}$ are k-bit integers).

Now $\mathcal{F}$ starts Phase 2, the adaptive chosen message attack, by running $\mathcal{A}$'s attack. When $\mathcal{A}$ asks for message M to be signed, the forger $\mathcal{F}$ asks its own signature oracle for a signature on M, therefore receiving the values σ, σ' such that $\sigma^e = (\sigma')^{e'} = y = H(M) \bmod N$. Notice that σ is the signature on M that must be computed in the simulation of TFDH-RSA.

To complete the simulation for $\mathcal{A}$, the forger $\mathcal{F}$ must now use σ to produce the signature fragments of the good parties. Let $f(x)$ be the polynomial (modulo m) of degree t that satisfies $f(i) = d_i$ for every $i \in B$ and $f(0) = d$. Then, the value of σ_j for $j \notin B$ is

$$\sigma_j = y^{2^{kt} f(j)} \bmod N$$

Remember that

$$f(j) = \sum_{i \in \tilde{B}} L_{\tilde{B}}(j, i) \cdot f(i) \bmod m$$

but because the values $L_{\tilde{B}}(j, i)$ are fractions we cannot compute this value directly or in the exponent. But by multiplying the sum by $\Gamma_{\tilde{B}}$ will remove all the denominators. By using σ' the forger can also bypass the problem of taking $e(\tilde{B})$-roots:

Using Shamir's method of "GCD in the exponent", the forger $\mathcal{F}$ first computes a value w such that $w^{e' \cdot e} = y \bmod N$. Namely, since $GCD(e, e') = 1$ then $\mathcal{F}$ can find integers a, b such that $ae + be' = 1$, and setting $w = \sigma^b (\sigma')^a \bmod N$ yields the required value. Next, for each $i \in \tilde{B}$ the forger $\mathcal{F}$ computes the integer $\lambda_{j,i} \stackrel{\text{def}}{=} \Gamma_{\tilde{B}} \cdot L_{\tilde{B}}(j, i)$. (These are indeed integers since the denominator in each of the Lagrangian coefficients divides $\Delta_{\tilde{B}}$ and therefore also $\Gamma_{\tilde{B}}$.) Finally, the forger computes $\sigma_j = w^{\lambda_{j,0}} \cdot \prod_{i \in B} (\sigma')^{\lambda_{j,i} d_i} \pmod N$. To see that this is the correct value, observe that:

$$\begin{aligned} w^{\lambda_{j,0}} \cdot \prod_{i \in B} (\sigma')^{\lambda_{j,i} d_i} &= w^{\Gamma_{\tilde{B}} L_{\tilde{B}}(j,0)} \cdot \prod_{i \in B} (\sigma')^{\Gamma_{\tilde{B}} L_{\tilde{B}}(j,i) d_i} \\ &\overset{(a)}{=} w^{(2^{kt} e') L_{\tilde{B}}(j,0)} \cdot \prod_{i \in B} (\sigma')^{(2^{kt} e') L_{\tilde{B}}(j,i) d_i} \\ &\overset{(b)}{=} (w^{2^{kt}})^{d \cdot e \cdot e' L_{\tilde{B}}(j,0)} \cdot \prod_{i \in B} (\sigma')^{2^{kt} e' L_{\tilde{B}}(j,i) d_i} \\ &= (w^{e' \cdot e})^{2^{kt} L_{\tilde{B}}(j,0) d} \cdot \prod_{i \in B} ((\sigma')^{e'})^{2^{kt} L_{\tilde{B}}(j,i) d_i} \\ &= \prod_{i \in \tilde{B}} y^{2^{kt} L_{\tilde{B}}(j,i) f(i)} \quad = \quad y^{2^{kt} \sum_{i \in \tilde{B}} L_{\tilde{B}}(j,i) f(i)} \\ &= y^{2^{kt} f(j)} \quad = \quad \sigma_j \pmod N \end{aligned}$$

where Equality (a) holds since $2^{kt} e' = 2^{kt} e(\tilde{B}) = \Gamma_{\tilde{B}}$, and Equality (b) holds since $w^{2^{kt}}$ is a quadratic residue modulo N and hence its order divides m, and since $d = e^{-1} \bmod m$ then $(w^{2^{kt}})^{d \cdot e} = w^{2^{kt}} \bmod N$. (Note that since we set $d = e^{-1} \bmod m$ and not $d = e^{-1} \bmod \lambda(N)$, then $z^{de} = z$ does not necessarily hold when z is not a quadratic residue modulo N.)

It follows from the description above that the simulated view that $\mathcal{A}$ sees in this run of $\mathcal{F}$ is almost identical to its view in the interaction with the protocol (the only difference is the negligible difference in the distribution of the d_{i_j}'s). Hence with probability negligibly close to ϵ, $\mathcal{A}$ outputs a valid forgery $\hat{\sigma}$ on some message $\hat{M}$. Namely, $\hat{\sigma}^e = H(\hat{M}) \bmod N$. The forger $\mathcal{F}$ then chooses an arbitrary value $\hat{\sigma}' \in Z_N^*$ and the pair $(\hat{M}, \langle \hat{\sigma}, \hat{\sigma}' \rangle)$ is a valid forgery for D-RSA (recall that the second "signature" is not verified in D-RSA).

Remark. Note the reason for separating the powers of two in $\Delta_{\tilde{B}}$ from the odd factors: For the assumption that we make about security of D-RSA, it is crucial that the parameter e' be odd (since letting the adversary to extract even roots might allow it to factor the modulus N). Hence we must limit the extra help that the simulator can get to only odd roots, and the even factors must be handled in the protocol itself.

Theorem 2. [illegible] H [illegible] t [illegible]

One way to prove Theorem 2 is to observe that under the RSA assumption, the D-RSA signature scheme is two-phase secure in the random-oracle model (where the proof is nearly identical to the security proof for Full-Domain-Hash signatures), and so we can use Theorem 1 from above.

5 Dynamic Additions of New Parties

We consider highly dynamic networks, where parties can join at any time and must be provided with shares of the signature key when they join. In some cases it may be reasonable to assume that such a share is installed by a trusted entity before the party is added to the network, but in other, less centralized situations, the parties already deployed will have to cooperate in order to furnish the new party with a share of the signing key.

In [25] an elegant method was introduced to solve the problem of player addition in secret sharing. The advantage of their scheme is that it is non-interactive, that is, each of the original $t+1$ parties only needs to send a single message to the new party P_{new}. The key idea of their approach is to use a bivariate polynomial. Similarly to the original Shamir's scheme, their scheme works over a prime field.

In this section we show how to adapt their scheme to work for the purpose of a threshold RSA scheme. The technical problems are the same at the ones we saw in the previous sections (the interpolation happens modulo a secret number m). We use some similar techniques to those used earlier, but the end result is somewhat different. When adding a new party P_{new} to the network, the new share given to it is not the same share as the one it would have received from the dealer in the sharing phase, but rather some multiple of that original secret. Thus, we need to modify the signature computation phase to incorporate these different shares. We show in the following that we are still able to generate signatures in a threshold fashion.

We modify our description of the threshold RSA from the previous sections so that the sharing of the secret is done via a bivariate polynomial. We note that if we do not consider newly added shares then this modification only affects the format in which the shares are represented but leaves the signature generation protocol exactly as it is in the case of the single-variate polynomial. The details follow. Again, we describe the protocol in the case of a honest-but-curious adversary. The details to add robustness (security against a malicious adversary) appear in Section 6.[2]

Sharing Phase: Given the public key N, e and the secret key d (such that $d = e^{-1} \pmod{m}$), the dealer chooses $(t+1)^2$ values $a_{i,j} \in Z_m$ (for $i, j \in [0, t]$), at random subject to $a_{i,j} = a_{j,i}$ and $a_{0,0} = d$. The dealer then defines the polynomial $f(x, y) = \sum_{i,j} a_{i,j} x^i y^j$. Note that since $a_{i,j} = a_{j,i}$ then the polynomial is symmetric, $f(x, y) = f(y, x)$. The share of party i is the polynomial $d_i(x) = f(x, i)$. (We note that only the free term $d_i(0) = f(0, i)$ will be used when computing the signature fragments, so reconstructing the signature from the fragments is unchanged from before.)

[2] We note that performing secret sharing via a bivariate polynomial has the added advantage, that it provides a check that the dealer is actually sharing a unique secret. This is called *verifiable secret sharing* (VSS) [1]. We do not discuss this feature in this paper, but we point out that it could be useful in some applications.

For the design of our protocol we also need to define an additional value δ_i. Initially, for the parties who receive shares from the dealer we have that $\delta_i = 1$. Thus each party P_i holds a polynomial $d_i(x)$ and an integer δ_i. The system invariant that we maintain is that for every party i, $d_i(x) \equiv \delta_i f(x,i) \bmod m$. Note that this trivially holds for all the original parties (for which $\delta_i = 1$).

Incorporating a New Party: When a new party P_{new} joins the group, it needs to receive its shares $d_{new}(x)$ and δ_{new}, while maintaining the system invariant that $d_{new}(x) \equiv \delta_{new} f(x, new) \bmod m$. This is done by having party P_{new}:

1. Receive from every P_{i_j}, $j = 1, \ldots, t+1$, the values $\alpha_{i_j} = d_{i_j}(new)$ and δ_{i_j}.
2. Let $\delta = \text{lcm}(\delta_{i_1}, \ldots, \delta_{i_{t+1}})$ and recall the definition of Δ_S (Eq. 3) for the set $S = \{i_1, \ldots, i_{t+1}\}$. Compute its share as:

$$d_{new}(x) = \delta \Delta_S f_{new}(x) = \sum_{j=1}^{t+1} \Delta_S L_S(x, i_j) \frac{\delta}{\delta_{i_j}} \alpha_{i_j} \tag{7}$$

3. Stores as its share this interpolated polynomial $d_{new}(x)$ and the value $\delta_{new} = \delta \cdot \Delta_S$.

To see that this computation gives the new party its appropriate share preserving the invariant and that it is feasible to compute this value consider the following. By definition $f_{new}(x) = f(x, new)$. Note that due to the symmetry of $f(x,y)$ and the system invariant, if P_i is a party already in the network, then

$$f_{new}(i) = f(i, new) = f(new, i) = \frac{d_i(new)}{\delta_i} \bmod m \tag{8}$$

though we can't explicitly compute the last fraction mod m.

Given $\delta = \text{lcm}(\delta_{i_1}, \ldots, \delta_{i_{t+1}})$ as defined in the protocol, multiplying both sides of Eq.8 by δ and specifying $i = i_j$ for all $j = 1, \ldots, t+1$ we have:

$$\delta f_{new}(i_j) = \frac{\delta}{\delta_{i_j}} d_i(new) \bmod m$$

Notice that now the fraction is an integer and can be computed even without knowing m.

Given the values $\alpha_{i_j} = d_{i_j}(new)$ and δ_{i_j} for $j = 1, \ldots, t+1$, we can interpolate the polynomial $\delta f_{new}(x)$ as

$$\delta f_{new}(x) \stackrel{\text{def}}{=} \sum_{j=1}^{t+1} L_S(x, i_j) \frac{\delta}{\delta_{i_j}} \alpha_{i_j} \bmod m \tag{9}$$

where $L_S(x, i_j)$ is the appropriate Lagrangian coefficient (see Eq. 1). Yet, as this computation needs to be computed mod m and requires the calculation of the Lagrangian coefficients mod m this cannot be done directly. To enable the computation we employ the techniques described in the previous sections.

We multiply Eq.(9) by Δ_S resulting in the computation of Eq. (7). Notice that multiplying by Δ_S removes all the denominators on the right-hand side of the equation. Furthermore, setting $\delta_{new} = \delta \cdot \Delta_S$ preserves the system invariant.

Signature Computation Phase: Assume that we want to compute the signature σ on a message M. Let $y = H(M)$, then we have $\sigma = y^d \bmod N$.

Recall that each party P_i holds the polynomial $d_i(x)$ and the integer δ_i such that $d_i(x) = \delta_i f(x, i) \bmod m$. Thus each party P_i publishes as its signature fragment the pair $(\sigma_i = y^{2^{kt} d_i(0)}, \delta_i)$.

Given $t+1$ of these signature fragments published by parties $P_{i_1}, \ldots, P_{i_{t+1}}$, δ and Δ_S as above, set

$$\sigma' = \prod_{j=1}^{t+1} \sigma_{i_j}^{\frac{\delta}{\delta_{i_j}} \cdot \Delta_S \cdot L_S(0, i_j)} \bmod N \tag{10}$$

Let $e' = 2^{kt} \cdot \delta \cdot \Delta_S$, compute integers a, b such that $ae + be' = 1$. Set the signature to:

$$\sigma = y^a \sigma'^b \bmod N$$

We show that the above computation generates a proper signature. First note that all the exponents in Eq. (10) are integers (as Δ_S removes the denominators from the Lagrangians, and δ_{i_j} divides δ by definition). Furthermore,

$$\sigma' = y^{\sum_{j=1}^{t+1} 2^{kt} \cdot \frac{\delta}{\delta_{i_j}} \cdot \Delta_S \cdot L_S(0, i_j) d_{i_j}(0)} \bmod N \tag{11}$$

Let's focus on the exponent of y in the above equation: by using the invariant $d_{i_j}(0) = \delta_{i_j} f(0, i_j)$ we have that the exponent equals:

$$\sum_{j=1}^{t+1} 2^{kt} \cdot \delta \cdot \Delta_S \cdot L_S(0, i_j) f(0, i_j)$$

From the polynomial interpolation we have that $\sum_{j=1}^{t+1} L_S(0, i_j) f(0, i_j) = f(0, 0) = d$ therefore

$$\sigma' = y^{2^{kt} \cdot \delta \cdot \Delta_S \cdot d} \bmod N \tag{12}$$

Assuming that $GCD(e, e') = 1$ we use the techniques from the previous section to extract the signature σ out of σ'. Indeed $\sigma' = \sigma^{e'} \bmod N$ and there exists integers a, b such that $ae + be' = 1$. Therefore

$$\sigma = \sigma^{ae+be'} = y^a \sigma'^b \bmod N$$

We remark that if we choose e as a prime larger than 2^k (the largest possible identity), then $GCD(e, e')$ is guaranteed to be 1 as the value e' can only be the product of differences of identities.

The proofs follow directly from the proofs of the previous section while incorporating the extra factor δ in the simulation.

Remark: Notice that the shares held by parties in this modified scheme are larger. Just because of the bivariate polynomial technique, each party holds $t+1$ values mod m rather than a single one. Moreover the size of the shares of parties added later in the system grow with the additive factor $\log \delta + \log \Delta_S$ (notice that Eq. (7) is computed over the integers by P_{new}).

Remark: Fazio [illegible] in [12] consider Shoup's original protocol and show how to add parties without using bivariate polynomials. Their work is not directly applicable to "ad hoc" networks as it requires [illegible] parties in the network to participate in assigning a share to a new party. On the other hand their solution does not increase the share size by a factor of t and may have more enhanced properties, such as proactive security.

6 Adding Robustness

The protocols described in the previous sections work only in the presence of an honest-but-curious adversary. Here we show how to tolerate a malicious adversary.

During the sharing phase, the dealer chooses a random value $g \in Z_N^*$ (with high probability g has order m) and publishes the values $G_{i,j} = g^{a_{i,j}} \bmod N$ for all the coefficients $a_{i,j}$ of the sharing polynomial f. Notice that this allows any party to compute $g^{f(i,j)}$ on any point (i,j) by "polynomial evaluation in the exponent".

When a new party P_{new} joins the network, it receives from an existing party P_i the values δ_i and $\alpha_i = d_i(new) = \delta_i f(new, i)$. Then P_{new} checks that $g^{\alpha_i} = [g^{f(new,i)}]^{\delta_i} \bmod N$ where $g^{f(new,i)}$ is computed using the values $G_{i,j}$ published by the dealer.

When computing a signature on a message M, where $y = H(M)$, a party P_i publishes the values δ_i and $\sigma_i = y^{2ktd_i(0)} \bmod N$, where $d_i(0) = \delta_i f(0,i)$ and proves that

$$\log_g [g^{f(0,i)}]^{\delta_i} = \log_{y^{2kt+1}} \sigma_i^2 \bmod m$$

(the extra squaring operation is needed to make sure that we are in the subgroup of order m in Z_N^*). Only signature fragments that pass the above verification test will be accepted.

Efficient zero-knowledge proofs for this language were presented in [19] and [18]. The ZK proof presented in [18] can be made non-interactive using the Fiat-Shamir heuristic in the random oracle model.

Acknowledgment. We are grateful to Stas Jarecki and the anonymous Eurocrypt reviewers for their helpful comments. Research was sponsored by US Army Research laboratory and the UK Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of

Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations. In: Proc. 20th STOC, pp. 1–10. ACM Press, New York (1988)
2. Blakley, G.R.: Safeguarding cryptographic keys. In: Proc. AFIPS 1979 National Computer Conference, AFIPS, pp. 313–317. (1979)
3. Boyd, C.: Digital Multisignatures. In: Baker, H., Piper, F. (eds.) Cryptography and Coding, pp. 241–246. Claredon Press, Oxford (1989)
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: 1st ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
5. Croft, R.A., Harris, S.P.: Public-key cryptography and re-usable shared secrets. In: Baker, H., Piper, F. (eds.) Cryptography and Coding, pp. 189–201. Claredon Press, Oxford (1989)
6. Cerecedo, M., Matsumoto, T., Imai, H.: Efficient and secure multiparty generation of digital signatures based on discrete logarithms. IEICE Trans. Fundamentals E76-A(4), 532–545 (1993)
7. Coron, J.-S.: On the Exact Security of Full Domain Hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
8. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: Proc. 26th STOC, pp. 522–533. ACM Press, New York (1994)
9. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 120–127. Springer, Heidelberg (1988)
10. Desmedt, Y.G.: Threshold cryptography. European Transactions on Telecommunications 5(4), 449–457 (1994)
11. Desmedt, Y., Frankel, Y.: Shared generation of authenticators and signatures. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 457–469. Springer, Heidelberg (1992)
12. Fazio, N.: Personal Communication (September 2007)
13. Frankel, Y., Gemmell, P., Mackenzie, P., Yung, M.: Proactive RSA. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 440–454. Springer, Heidelberg (1997)
14. Frankel, Y., Gemmell, P., Yung, M.: Witness-based Cryptographic Program Checking and Robust Function Sharing. In: Proc. 28th STOC, pp. 499–508. ACM Press, New York (1996)
15. Gennaro, R., Halevi, S., Krawczyk, H., Rabin, T.: Threshold RSA for Dynamic and Ad-Hoc Groups. Full version, http://eprint.iacr.org/2008/045
16. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust and efficient sharing of RSA functions. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 157–172. Springer, Heidelberg (1996)
17. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust Threshold DSS Signatures. Information and Computation 164(1), 54–84 (2001); Extended abstract in EUROCRYPT 1996

18. Gennaro, R., Krawczyk, H., Rabin, T.: RSA-based Undeniable Signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 397–416. Springer, Heidelberg (1997); Final version in J. Cryptology 13(4), 397–416 (2000)
19. Gennaro, R., Krawczyk, H., Rabin, T.: Robust and Efficient Sharing of RSA Functions. Journal of Cryptology 13(2), 273–300 (2000); Conference version [16]
20. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Computing 17(2), 281–308 (1988)
21. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communication of the ACM 21(2), 120–126 (1978)
22. Shamir, A.: How to Share a Secret. Communications of the ACM 22, 612–613 (1979)
23. Shamir, A.: On the Generation of Cryptographically Strong Pseudorandom Sequences. ACM Trans. Comput. Syst. 1(1), 38–44 (1983)
24. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
25. Saxena, N., Tsudik, G., Yi, J.H.: Efficient node admission for short-lived mobile ad hoc networks. In: ICNP, 13th IEEE International Conference on Network Protocols, pp. 269–278 (2005)

Towards Key-Dependent Message Security in the Standard Model

Dennis Hofheinz[1] and Dominique Unruh[2]

[1] CWI, Cryptology and Information Security Group, Amsterdam, The Netherlands
Dennis.Hofheinz@cwi.nl
[2] Saarland University, Information Security and Cryptology Group, Saarbrücken, Germany
unruh@cs.uni-sb.de

Abstract. Standard security notions for encryption schemes do not guarantee any security if the encrypted messages depend on the secret key. Yet it is exactly the stronger notion of security in the presence of *key-dependent* messages (KDM security) that is required in a number of applications: most prominently, KDM security plays an important role in analyzing cryptographic multi-party protocols in a formal calculus. But although often assumed, the mere existence of KDM secure schemes is an open problem. The only previously known construction was proven secure in the random oracle model.

We present symmetric encryption schemes that are KDM secure in the standard model (i.e., without random oracles). The price we pay is that we achieve only a relaxed (but still useful) notion of key-dependent message security. Our work answers (at least partially) an open problem posed by Black, Rogaway, and Shrimpton. More concretely, our contributions are as follows:

1. We present a (stateless) symmetric encryption scheme that is information-theoretically secure in face of a *bounded* number and length of encryptions for which the messages depend in an arbitrary way on the secret key.
2. We present a stateful symmetric encryption scheme that is computationally secure in face of an arbitrary number of encryptions for which the messages depend only on the respective *current* secret state/key of the scheme. The underlying computational assumption is minimal: we assume the existence of one-way functions.
3. We give evidence that the only previously known KDM secure encryption scheme cannot be proven secure in the standard model (i.e., without random oracles).

Keywords: Key-dependent message security, security proofs, symmetric encryption schemes.

1 Introduction

Proofs of security are a good and sound way to establish confidence in an encryption system. However, "proof" is a bit misleading here: usually, a security proof

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 108–126, 2008.

is not an absolute statement, but merely shows that [illegible], the scheme is resistant against a [illegible]. Nothing is guaranteed if the assumptions are invalidated or attacks outside the considered class take place. Therefore, it is crucial that

– the underlying assumptions are plausible, and
– the considered class of attacks is as general as possible.

Additionally, encryption schemes are most often used only as a building block in a larger protocol context, and thus

– the considered class of attacks should allow for meaningful and general analysis of the encryption scheme in a larger protocol context.

Indistinguishability of ciphertexts. The most established class of attacks consists of attacks targeted against the [illegible] (IND-CPA [16], resp. IND-CCA [21] attacks). Here, adversary A's goal is to win the following game: first, A chooses two messages m_0, m_1, then gets the encryption c_b of m_b (for a random $b \in \{0, 1\}$), and finally outputs a guess b' for b. Now A wins if $b = b'$, i.e., if it guessed correctly which message was encrypted. The scheme is secure if no adversary wins (significantly) more often than in half of the cases. Intuitively, security in this sense implies that "one ciphertext looks like any other."

The IND-CPA and IND-CCA notions have been tremendously successful and even proved equivalent to a number of alternative and arguably not less appealing notions (cf. [5,6,10,19]). At the same time, IND-CPA and IND-CCA security can be achieved under various plausible number-theoretic assumptions [16,13,11].

Key-dependent message security. However, there is one security property that is useful and important in many applications, yet is [illegible] covered by IND-CPA or IND-CCA security: security in presence of [illegible] messages. More concretely, imagine a scenario in which the adversary can request encryptions of [illegible]. In other words, the adversary chooses a function g and gets the encryption of $g(K)$ under secret key K. Note that this is something the adversary may not be able to generate on its own, not even in the public-key setting. The adversary's goal is now to distinguish such a key-dependent encryption from an encryption of a random message. Security of an encryption is a [illegible] notion to consider since

– in relevant practical settings, this notion is necessary: consider, e.g., encrypting your hard drive (which may contain the secret key, e.g., on the swap partition, or in a file that contains your secret keyring),
– certain protocols use key-dependent message security explicitly as a technical tool [8],

and, possibly most importantly from a theoretical perspective,

– key-dependent message security is a key ingredient for showing that security results that are proven in a formal calculus are also computationally sound.

This latter reason may come a bit surprising, hence we explain it in more detail.

Formal security proofs. The idea to automate security proofs can be traced back to the seminal work of Dolev and Yao [14], who described a formal calculus to analyze security protocols. To make the calculus accessible to automatic provers, however, base primitives like encryption (or, later, signatures) had to be over-idealized, disconnecting them from their concrete computational implementations. What was missing for almost 20 years was a soundness result, i.e., a result that essentially states "whatever can be proven in the abstract calculus holds as well in the cryptographic world, where the ideal encryption operator is implemented with an encryption scheme."

But finally, the soundness result by Abadi and Rogaway [1] connected the formal, machine-accessible world with the cryptographic world. However, with standard encryption schemes, only a certain subset of possible protocols could be considered, namely those that only contain expressions which fulfil a certain "acyclicity" condition.[1] To achieve full generality, a stronger requirement (security in the presence of key-dependent messages) on the encryption scheme was needed. This is not a peculiarity of the approach of Abadi and Rogaway; similar problems occur in related approaches, e.g. [20,2,4]. In particular, Adão et al. [2] show that in a certain sense, key-dependent message security is a necessity for formal soundness.

Related work. Around the time when the need for key-dependent security had been realized, formal characterizations of the security notion were given in [8,7]. Moreover, [7] showed a simple symmetric encryption scheme to be secure with respect to their notion. However, their scheme was proven in the random oracle model, and the proof made heavy use of the "ideal" nature of the random oracle (more details on this in Section 3). Black et al. posed the question of achieving key-dependent security in the [illegible] model.

Backes et al. [3] consider several strengthenings of the definition from [7]. They prove structural results among the notions (including a way to "patch" a scheme that is secure in the sense of [7] to match the notions from [3]). However, Backes et al. do not give an actual construction of a secure scheme.

Our work. Our goal is to achieve key-dependent message security, as defined by Black et al., in the standard model. We present several results:

- a (stateless) symmetric encryption scheme that is information-theoretically secure in face of a [illegible] number and length of encryptions for which the messages depend in an arbitrary way on the secret key.
- a stateful symmetric encryption scheme that is computationally secure in face of an arbitrary number of encryptions for which the messages depend only on the respective [illegible] secret state/key of the scheme. The underlying computational assumption is minimal: we assume the existence of one-way functions.

[1] They also did only prove security against passive adversaries. However, active security was achieved by subsequently by [20,2,4].

We also stress the strictness of key-dependent message security:

– We give evidence that the only previously known KDM secure encryption scheme cannot be proven secure in the standard model (i.e., without random oracles).[2]

Note. Recently, we learned about the (concurrent and independent) work [17] of Halevi and Krawczyk. They are interested more generally in keyed primitives (such as pseudorandom functions, PRFs) which are secure in face of key-dependent inputs. They also show that an encryption scheme constructed from such a PRF inherits the underlying PRF's resilience against key-dependent inputs/messages. In particular, Halevi and Krawczyk construct a PRF (and a corresponding encryption scheme) that is secure in face of inputs which depend in an arbitrary, but [illegible] way on the key. (That is, for each way in which the query may depend on the key, they give a PRF which is secure in face of such inputs.)

In contrast to that, we are interested in constructing encryption schemes that are secure in face of (encryptions of) messages that depend in an [illegible], adaptively determined way on the key. Unfortunately, neither our schemes nor the schemes of [17] can handle the important case of non-trivial [illegible], that is, cyclic chains of encryptions of key K_i under key $K_{i+1 \bmod n}$.

2 Preliminaries

Basic notation. Throughout the paper, $k \in \mathbb{N}$ denotes the [illegible] of a given construction. Intuitively, a larger security parameter should provide more security, but a scheme's efficiency is also allowed to degrade with growing k. A [illegible] function vanishes faster than any given polynomial. The [illegible] between two random variables X and Y is denoted by $\delta(X\,;\,Y)$. The [illegible] $H_2(X)$ of a random variable X is defined as $H_2(X) := -\sum_x \log_2 \mathsf{Pr}\,[X = x]^2$. Two families (X_k) and (Y_k) of random variables are [illegible] (written $X \approx Y$) if for every [illegible] (probabilistic polynomial-time) algorithm A, the function $|\mathsf{Pr}\,[A(X_k) = 1] - \mathsf{Pr}\,[A(Y_k) = 1]|$ is negligible in k. A family $\mathcal{UHF}$ of [illegible] is a family of functions $h : \{0,1\}^n \to \{0,1\}^m$ with the property that for $x, x' \in \{0,1\}^n$ with $x \neq x'$, all $y, y' \in \{0,1\}^m$, and uniformly chosen $h \in \mathcal{UHF}$, we have that $\Pr[h(x) = y, h(x') = y'] = 2^{-2m}$.

We will further need a strengthened version of the leftover hash lemma that takes into account additional information S about the randomness K and some additional information Q unrelated to K.

Lemma 1 (Leftover Hash Lemma, extended). [illegible] K [illegible] Q [illegible] S [illegible] h [illegible] $\mathcal{UHF}$ [illegible] U [illegible] $|h(K)|$ [illegible]

[2] A similar, but technically different result is also contained in the independent work [17].

– U [illegible] (h, S, Q) [illegible]
– K [illegible] Q [illegible]
– h [illegible] (K, S, Q) [illegible]

[illegible]

$$\delta(h, h(K), S, Q \ ; \ h, U, S, Q) \ \leq \ 2^{|S|+|h(K)|/2-H_2(K)/2-1}.$$

In a typical application of this lemma, h, K, and Q would be mutually independent, and S would be a function of (h, K, Q) (say, a side channel). Furthermore, U would be some completely independent random variable, representing the ideal randomness. This would then imply all the independence conditions in the lemma.

[illegible] In the following, s, q, k range over all values taken by S, Q, K, respectively. By applying the definition of the statistical distance, we have

$$\begin{aligned} \varepsilon &:= \delta(h, h(K), S, Q \ ; \ h, U, S, Q) \\ &= \sum_{s,q} \Pr[S = s, Q = q] \, \delta(h, h(K)|S = s, Q = q \ ; \ h, U|S = s, Q = q). \end{aligned} \tag{1}$$

Here $X|(S = s)$ stands for the distribution of X under the condition $S = s$. Since h and (K, S, Q) are independent, $h|(S = s, Q = q)$ is a universal hash-function. And since U is independent of (S, Q, h), we have that U is uniformly distributed and independent of h given $S = s, Q = q$. Further, since by assumption h is independent of (K, S, Q), we have that h and K are independent given $S = s, Q = q$. Thus the leftover hash lemma in its basic form [18] applies, and we get

$$\delta(h, h(K)|S = s, Q = q \ ; \ h, U|S = s, Q = q) \ \leq \ 2^{|h(K)|/2-H_2(K|(S=s,Q=q))/2-1}.$$

Combining this with (1) we get

$$\begin{aligned} \varepsilon &\leq \sum_{s,q} \Pr[S = s, Q = q] \cdot 2^{|h(K)|/2-H_2(K|(S=s,Q=q))/2-1} \\ &= \sum_{s,q} \Pr[S = s, Q = q] \cdot \frac{1}{2}\sqrt{2^{|h(K)|} \cdot \sum_k \Pr[K = k|S = s, Q = q]^2} \\ &\leq \sum_{s,q} \Pr[Q = q] \cdot \frac{1}{2}\sqrt{2^{|h(K)|} \cdot \sum_k \Pr[S = s|Q = q]^2 \cdot \Pr[K = k|S = s, Q = q]^2} \\ &= \sum_{s,q} \Pr[Q = q] \cdot \frac{1}{2}\sqrt{2^{|h(K)|} \cdot \sum_k \Pr[K = k, S = s|Q = q]^2} \\ &\leq \sum_{s,q} \Pr[Q = q] \cdot \frac{1}{2}\sqrt{2^{|h(K)|} \cdot \sum_k \Pr[K = k|Q = q]^2} \end{aligned}$$

$$\overset{(*)}{=} \sum_{s,q} \Pr[Q=q] \cdot \frac{1}{2}\sqrt{2^{|h(K)|} \cdot \sum_k \Pr[K=k]^2}$$
$$= \sum_{s,q} \Pr[Q=q] \cdot \frac{1}{2}\sqrt{2^{|h(K)|} \cdot 2^{-H_2(K)}}$$
$$= \sum_{s,q} \Pr[Q=q] \cdot 2^{|H(k)|/2 - H_2(K)-1}$$
$$= \sum_{s} 2^{|H(k)|/2 - H_2(K)-1} = 2^{|S|+|H(k)|/2-H_2(K)-1}.$$

Here $(*)$ uses that Q and K are independent. □

Key-dependent message security. For formalizing key-dependent message security, we use a variation on the definition of Black et al. [7]:

Definition 2 (KDM security, standard model, symmetric setting). [illegible] $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ [illegible] $\mathbf{K} := (K_1, \ldots, K_n)$ [illegible] ([illegible] n [illegible]) [illegible] A [illegible]

– $\mathsf{Real}_{\mathbf{K}}$ [illegible] g, μ [illegible] $C \leftarrow \mathcal{E}(1^k, K_\mu, g(\mathbf{K}))$ [illegible]
– $\mathsf{Fake}_{\mathbf{K}}$ [illegible] g, μ [illegible] $C \leftarrow \mathcal{E}(1^k, K_\mu, U)$ [illegible] $U \in \{0,1\}^{|g(\mathbf{K})|}$

[illegible] g [illegible] [3] [illegible] KDM advantage [illegible] A [illegible]

$$\mathsf{Adv}_{\Pi}^{KDM}(A) := \left|\Pr\left[\mathbf{K} \xleftarrow{\$} \mathcal{K} : A^{\mathsf{Real}_{\mathbf{K}}(\cdot)} = 1\right] - \Pr\left[\mathbf{K} \xleftarrow{\$} \mathcal{K} : A^{\mathsf{Fake}_{\mathbf{K}}(\cdot)} = 1\right]\right|$$

[illegible] $\mathbf{K} \xleftarrow{\$} \mathcal{K}$ [illegible] K_i [illegible] $\mathcal{K}$ [illegible] Π [illegible] KDM secure [illegible] A [illegible] n [illegible] $\mathsf{Adv}_{\Pi}^{KDM}(A)$ [illegible] A [illegible] g [illegible] $|g(K)|$ [illegible] K [illegible]

The relation to real-or-random security. Definition 2 bears a great resemblance to the real-or-random (ROR-CPA) definition for encryption schemes from [5]. The main difference is that Definition 2 equips the adversary with an oracle that delivers encryptions of [illegible] messages (i.e., evaluations) $g(K)$. The way in which these messages depend on the keys is completely up to the adversary; the only constraint is that g must be efficiently evaluatable and have a fixed output length.

[3] This has the side-effect that for a polynomial-time adversary A, the function g is also polynomial-time computable.

On achieving KDM security and active KDM security. Using the equivalence of ROR-CPA and IND-CPA security from [5], it is easy to see that Definition 2 is [illegible] stronger than IND-CPA security. A natural adaption of Definition 2 to active attacks—such a notion is called AKDM security in [3]—consists in equipping the adversary with a decryption oracle that is restricted in the usual sense to prevent trivial attacks. And similarly to the passive case, it is easy to see that AKDM security is [illegible] stronger than IND-CCA security. On the other hand, once a scheme is KDM secure, it can be easily and without (much) loss of efficiency upgraded to AKDM security, as formalized and proved in [3]. Hence, the main difficulty lies in finding a scheme that is KDM secure in the first place. In the following, this will be our focus.

3 The Scheme of Black et al

Definition 2 is very hard to achieve. In fact, the only construction that is known, due to Black et al. [7], to achieve Definition 2 is in the random oracle model. It will be very useful to take a closer look at their scheme. We will argue that in a very concrete sense, nothing less than a random oracle will do for their scheme. Hence, their construction merely shows how powerful random oracles are, but does not give a hint on how to achieve KDM security in the standard model. This constitutes one motivation for our upcoming weakening of KDM security.

Scheme 3 (The scheme ver). Define the symmetric encryption scheme **ver** $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with security parameter $k \in \mathbb{N}$, message space $\{0,1\}^k$ and key space $\{0,1\}^k$ through

- $\mathcal{K}(1^k)$ outputs a uniform random key $K \in \{0,1\}^k$.
- $\mathcal{E}(1^k, K, M)$ samples $R \overset{\$}{\leftarrow} \{0,1\}^k$ and outputs the ciphertext $(R, H(K||R) \oplus M)$.
- $\mathcal{D}(1^k, K, (R, D))$ outputs the message $H(K||R) \oplus D$.

The security of ver with a random oracle. Black et al. prove

Theorem 4 (Security of ver [7]). [illegible] H [illegible] **ver** [illegible]

The main idea of the proof is to consider an event **bad**, where **bad** occurs iff

1. the adversary queries H at any point $K||R$ that was [illegible] used for encryption, or
2. one of the functions g submitted to the encryption oracle queries H at the [illegible] point $K||R$.

If **bad** does not occur, the adversary's view is [illegible] in the Real and Fake experiments, thanks to the fact that different random oracle queries $H(X), H(Y)$ $(X \neq Y)$ are statistically independent: each message is padded with [illegible] and message-independent randomness. Hence, by showing (with an inductive argument) that **bad** occurs only with small probability, [7] show the scheme **ver** KDM secure.

The insecurity of ver without a random oracle. Put informally, the proof of **ver** utilizes one essential property of the random oracle H: knowledge about arbitrary many values $H(Y_i)$ (with $Y_i \neq X$) does not yield [illegible] information about $H(X)$. This use of a random oracle as a provider of statistical independence is what makes the proof fail completely with any concrete hash function used in place of the random oracle. There is no hope for the proof strategy to succeed without random oracles. A little more formally, we can show that in the random oracle model, there exists a specific hash function H that has a number of generally very useful properties: H is collision-resistant, one-way, can be interpreted as a pseudorandom function (in a way compatible with **ver**), and H makes **ver** IND-CPA. [illegible] H makes **ver** completely insecure in the presence of key-dependent messages. Hence, there can be no fully black-box KDM security proof for **ver** that relies on these properties of H alone.

Theorem 5 (Insecurity of ver). [illegible] $\mathcal{O}$ [illegible] H [illegible]

1. H [illegible]
2. [illegible] $p(k) \in k^{\Theta(1)}$, H [illegible] $\{0,1\}^{p(k)}$ [illegible]
3. [illegible] $F_K(R) := H(K||R)$ [illegible] K [illegible]
4. [illegible] **ver** [illegible] H [illegible]
5. [illegible] **ver** [illegible] H [illegible]

[illegible] Assume for simplicity that the security parameter k is even. Say that the random oracle $\mathcal{O}$ maps arbitrary bitstrings to k-bit strings. Then denote by $\mathcal{O}_\ell(x)$ the first $k/2$ bits of $\mathcal{O}(x)$. Now consider the function $H : \{0,1\}^* \to \{0,1\}^k$ with

$$H(x) := \begin{cases} \mathcal{O}(x) & \text{for } |x| \neq 2k, \\ \mathcal{O}(x_\ell) \oplus (\mathcal{O}_\ell(x)||\mathcal{O}_\ell(\mathcal{O}_\ell(x))) & \text{for } x = x_\ell||x_r \text{ and } |x_\ell| = |x_r| = k. \end{cases}$$

We show the claimed properties for H:

1. H is collision-resistant. It is clear that collisions $H(x) = H(y)$ (with $x \neq y$) cannot be found efficiently if $x \neq 2k$ or $y \neq 2k$. So assume $x = x_\ell||x_r$ and $y = y_\ell||y_r$ for $|x_\ell| = |x_r| = |y_\ell| = |y_r| = k$. Collisions of this form imply $\mathcal{O}_\ell(x_\ell) \oplus \mathcal{O}_\ell(x) = \mathcal{O}_\ell(y_\ell) \oplus \mathcal{O}_\ell(y)$ and thus

$$\mathcal{O}_\ell(x_\ell) \oplus \mathcal{O}_\ell(y_\ell) = \mathcal{O}_\ell(x) \oplus \mathcal{O}_\ell(y). \tag{2}$$

If $x_\ell = y_\ell$, then this constitutes a collision in $\mathcal{O}_\ell$, so we may assume $x_\ell \neq y_\ell$. But the distributions of $\mathcal{O}_\ell$ on k-bit strings and on $2k$-bit strings are independent and both uniform. Hence, finding x and y to satisfy (2) requires a superpolynomial number of queries to $\mathcal{O}_\ell$ (resp. $\mathcal{O}$) with overwhelming probability.

2. H is one-way w.r.t. the uniform distribution on $\{0,1\}^k$. For $p(k) = 2k$, this follows from collision-resistance and the fact that H is compressing: Since the preimages of H are not unique, if we are able to find a preimage x' of $H(x)$ for random $x \in \{0,1\}^{2k}$, with noticeable probability we will have $x \neq x'$. This allows to find collisions efficiently. For details see [12]. For $p(k) \neq 2k$, this follows by definition of H and the fact that the random oracle is one-way.

3. $F_K(R) := H(K||R)$ **is a pseudorandom function.** Consider an adversary A that has oracle access to $\mathcal{O}$ and to F_K for uniformly chosen K. We denote A's i-th query to F_K by R_i. Without loss of generality, assume that A never asks for the same F_K evaluation twice, so the R_i are pairwise distinct. Furthermore, let $X_i := K||R_i$, and $Y_i := \mathcal{O}_\ell(K||R_i)$. We claim that A doesn't query $\mathcal{O}$ with K or [illegible] of the values X_i, Y_i, except with negligible probability.

We prove our claim inductively as follows. Let E_i denote the event that A queries $\mathcal{O}$ with a value that starts with K [illegible] to the i-th F_K query. Clearly, E_1 happens with exponentially small probability. So fix an $i \geq 1$. To complete our proof, it is sufficient to show that under condition $\neg E_i$, the probability for E_{i+1} to happen is bounded by a negligible function that does not depend on i.

Assume that $\neg E_i$ holds. That means that, given A's view up to and including the $(i-1)$-th F_K query, the key K is uniformly distributed among all k-bit values (or k-bit prefixes of $2k$-bit values) not yet queried by A. By the polynomiality of A, this means that, from A's point of view, K is uniformly distributed on an exponentially-sized subset of $0,1^k$. But this means that until the i-th F_K query, A has only an exponentially small chance to query one of K, X_j, Y_j $(j < i)$. Hence $E_{i+1} \mid \neg E_i$ happens only with exponentially small probability.

Summing up, A never queries $\mathcal{O}$ with K or any of the X_i, Y_i, except with negligible probability. Hence, F_K can be substituted with a truly random function without A noticing, and the claim follows.

4. **ver with** H **is IND-CPA.** Follows immediately from 3.

5. **ver with** H **is not KDM secure.** A successful KDM adversary A on **ver** is the following: A asks its encryption oracle for an encryption of $\mathcal{O}(K)$ (e.g., using g with $g(x) = \mathcal{O}(x)$ as input to the oracle). In the real KDM game, the ciphertext will be

$$(R, H(K||R) \oplus \mathcal{O}(K)) = (R, \mathcal{O}_\ell(K||R)||\mathcal{O}_\ell(\mathcal{O}_\ell(K||R))),$$

and hence of the form $(R, t||\mathcal{O}_\ell(t))$ for some t, which can be easily recognized by A. But in the fake KDM game, the ciphertext will have the form (R, U) for a uniformly and independently distributed U, which is generally not of the form $(R, t||\mathcal{O}_\ell(t))$. Hence, A can successfully distinguish real encryptions from fake ones. □

Halevi and Krawczyk's example. Halevi and Krawczyk give a different example of the "non-implementability" of **ver** (see [17, Negative Example 4]). They argue that the random oracle H in **ver** cannot be implemented with a PRF that is constructed from an ideal cipher using the Davies-Meyer transform. Their example has the advantage of being less artificial, while being formulated in the ideal cipher model.

4 Information-Theoretic KDM Security

Since key-dependent message security is very hard to achieve, we start with two simple schemes that do not achieve full KDM security, but serve to explain some important concepts.

4.1 The General Idea and a Simple Scheme (Informal Presentation)

First observe that the usual one-time pad $C = M \oplus K$ (where C is the ciphertext, M the message, and K the key) does [illegible] achieve KDM security. Encryption of $M = K$ results in an all-zero ciphertext that is clearly distinguishable from a random encryption. However, the slight tweak

$$C = (h, M \oplus h(K)) \quad (h \text{ independently drawn universal hash function})$$

[illegible] achieve a certain form of key-dependent message security: the pad $h(K)$ that is distilled from K is indistinguishable from uniform and independent randomness, even if h and some arbitrary (but bounded) information $M = M(K)$ about K is known. (When using suitable bitlengths $|K|$ and $|M|$, this can be shown using the leftover hash lemma [18].) So the encryption $M \oplus h(K)$ of one single message $M = M(K)$ looks always like uniform randomness. Hence the scheme is KDM secure in a setting where the encryption oracle is only used once (but on the other hand, information-theoretic security against unbounded adversaries is achieved).

4.2 A More Formal Generalization of the Simple Scheme

Of course, one would expect that by expanding the key, the scheme stays secure even after [illegible] (key-dependent) encryptions. This is true, but to show this, a hybrid argument and multiple applications of the leftover hash lemma are necessary. We formalize this statement now.

Scheme 6 (The scheme p-BKDM (for "p-bounded KDM")). Let $p \in \mathbb{Z}[k]$ be a positively-valued polynomial, let $\ell(k) := (2p(k) + 3)k$, and let $\mathcal{UHF}$ be a family of universal hash functions that map $\ell(k)$-bit strings to k-bit strings. Define the symmetric encryption scheme p-**BKDM** $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with security parameter $k \in \mathbb{N}$, message space $\{0,1\}^k$, and key space $\{0,1\}^{\ell(k)}$ through

- $\mathcal{K}(1^k)$ outputs a uniform random key $K \in \{0,1\}^{\ell(k)}$.
- $\mathcal{E}(1^k, K, M)$ samples $h \xleftarrow{\$} \mathcal{UHF}$ and outputs the ciphertext $C = (h, h(K) \oplus M)$.
- $\mathcal{D}(1^k, K, (h, D))$ outputs the message $h(K) \oplus D$.

Definition 7 (Bounded KDM security). [illegible] $p \in \mathbb{Z}[k]$ [illegible] Π [illegible] p-bounded KDM secure [illegible] $p(k)$ [illegible] Π [illegible] information-theoretically p-bounded KDM secure [illegible] $p(k)$ [illegible]

Theorem 8 (Bounded KDM security of p-BKDM). [illegible]
p **BKDM** [illegible] p [illegible]

[illegible] In the following, we abbreviate $x_i, .., x_j$ with $x_{i..j}$ for all variables x. Let n be the number of keys used. Let an adversary A be given that queries the encryption oracle at most $p(k)$ times. Without loss of generality we can assume the adversary to be deterministic (by fixing the random tape that distinguishes best) and that it performs exactly $p(k)$ queries. In the i-th encryption in the real experiment, let μ_i denote the index of the key that has been used, let h_i be the hash function chosen by the encryption function, let m_i be the message that is encrypted, and let c_i be the second component of the resulting ciphertext (i.e., (h_i, c_i) is the i-th ciphertext). Since the adversary is deterministic, m_i depends deterministically from the keys $K_{1,n}$ and the ciphertexts $c_{1..i-1}, h_{1..i-1}$, i.e., there are deterministic functions $\hat{f}_i$ with $m_i = \hat{f}_i(K_{1,n}, c_{1..i-1}, h_{1..i-1})$. Similarly, there are deterministic functions $\hat{\mu}_i$ such that $\mu_i = \hat{\mu}_i(c_{1..i-1}, h_{1..i-1})$.

Let U_i be independent uniformly distributed random variables on $\{0,1\}^k$ that are independent of all random variables defined above. Let

$$\varepsilon_i := \delta(h_{1..i}, c_{1..i} \ ; \ h_{1..i}, U_{1..i})$$

To show that the scheme is information-theoretically p-bounded KDM secure, i.e., that the adversary cannot distinguish the real and the fake experiment, it is sufficient to show that $\varepsilon_{p(k)}$ is negligible since the view of A can be deterministically computed from $h_{1..p(k)}, c_{1..p(k)}$.

Fix some $i \in \{1, \dots, p(k)\}$. Let $K := K_{\mu_i}$, $Q := h_{1..i-1}$, $S := (m_i, c_{1..i-1})$, $h := h_i$ and let U be uniformly distributed on $\{0,1\}^k$ and independent of (K, Q, S, h). The following conditions hold by construction:

- h is a universal hash function.
- U is uniformly distributed and independent of (h, S, Q).
- K and Q are independent.
- h is independent of (K, S, Q).

So the conditions for Lemma 1 are fulfilled and we have

$$\delta(h, h(K), S, Q \ ; \ h, U, S, Q) \leq 2^{|S|+|h(K)|/2-H_2(K)/2-1} = 2^{ik+k/2-\ell(k)/2-1} \leq 2^{-k}$$

and thus

$$\begin{aligned} &\delta(h_{1..i}, c_i, c_{1..i-1} \ ; \ h_{1..i}, U_i, c_{1..i-1}) \\ &\qquad \leq \delta(h_{1..i}, h_i(K_{\mu_i}), m_i, c_{1..i-1} \ ; \ h_{1..i}, U, m_i, c_{1..i-1}) \leq 2^{-k} \end{aligned} \tag{3}$$

Since (h_i, U_i) is independent of $(h_{1..i-1}, c_{1..i-1}, U_{1..i-1})$ by construction, from (4.2) we have $\delta(h_{1..i}, U_i, c_{1..i-1} \ ; \ h_{1..i}, U_i, U_{1..i-1}) = \varepsilon_{i-1}$ and hence using (3) and the triangle inequality for the statistical distance, we have

$$\varepsilon_i = \delta(h_{1..i}, c_i, c_{1..i-1} \ ; \ h_{1..i}, U_i, U_{1..i-1}) \leq 2^{-k} + \varepsilon_{i-1}.$$

Since $\varepsilon_0 = 0$, it follows that $\varepsilon_{p(k)} \leq p(k) \cdot 2^{-k}$ is negligible. □

4.3 Discussion

The usefulness of bounded KDM security. Our scheme p-**BKDM** can be used in any protocol where the total length of the encrypted messages does not depend on the length of the key. At a first glance, this restriction seems to defeat our purpose to be able to handle key cycles: it is not even possible to encrypt a key with itself. However, a closer inspection reveals that key dependent messages occur in two kinds of settings. In the first setting, a protocol might make explicit use of key cycles in its protocol specification, e.g., it might encrypt a key with itself (we might call this [illegible]). In this case, p-**BKDM** cannot be used. In the second setting, a protocol does not explicitly construct key cycles, but just does not exclude the possibility that—due, e.g., to some leakage of the key—some messages turn out to depend on the keys (we might call this [illegible]). In this case, the protocol does not itself construct key cycles (so the restriction of p-**BKDM** that a message is shorter than the key does not pose a problem), but only requires that [illegible] the protocol is still secure. But this is exactly what is guaranteed by p-**BKDM**. So for the—possibly much larger—class of protocols with unintentional key cycles the p-**BKDM** scheme can be used.

Multiple sessions of p-BKDM. Theorem 8 guarantees that even in the case of multiple sessions, the scheme p-**BKDM** is secure assuming that at most $p(k)$ encryptions are performed [illegible]. In some applications, especially if the number of sessions cannot be bounded in advance, one might need the stronger property that we may encrypt $p(k)$ messages [illegible]. Intuitively, we might argue that when we receive an encryption $(h, M \oplus h(K))$ of a message M, the entropy of the key K decreases by at most $|M \oplus h(K)|$ bits, but as long as enough entropy remains in K, we do not learn anything about M, and neither about the keys M depends on. This leads to the following conjecture:

[illegible] The scheme p-**BKDM** is KDM-secure if the adversary performs at most $p(k)$ encryptions under each key K_i. This holds even if different keys have different associated polynomials p_i (i.e., key K_i has length $O(p_i(k)k)$ and we encrypt p_i times under K_i).

Unfortunately, we do not know how to formally prove Conjecture 9. Formalizing the above intuition is not straightforward, since it is not clear how to alone define what it means that the entropy of a given key decreases while the entropy of the others does not. We leave this conjecture as an open problem.

Why encrypt *only* key-dependent messages? Definitions 2 and 7 give the adversary (only) access to an encryption oracle which encrypts arbitrary functions of the key (in contrast to [17] which additionally provides an encryption oracle for normal messages). In Definition 2, no generality is lost, since an ordinary encryption oracle can be emulated by choosing this function as a constant function. Call such "ordinary" encryption queries [illegible]. Now it is conceivable that a scheme allows for an unbounded number of non-KDM queries, but only a limited number of actually key-dependent queries. The security of such schemes can be appropriately captured using, e.g., the security definition of [17], which incorporates encryption oracles for key-dependent and non-KDM queries. While our Definition 7 does not allow to model such schemes, it is easy to see that our scheme p-**BKDM** is not secure against an unbounded number of non-KDM encryptions (not even against computationally bounded adversaries).

5 Computational KDM Security

5.1 Motivation

The dilemma with hybrid arguments. The discussion in Section 4.3 does not only apply to our scheme p-**BKDM**. There seems to be a general problem with proving KDM security with a hybrid argument. Starting with the real KDM game, substituting the first encryption with a fake one first is not an option: the later encryptions cannot be properly simulated. But to substitute the last real encryption first is not easy either: for this, there first of all has to be a that at that point, the last key has not already leaked completely to the adversary. In our case, with a bounded overall number of encryptions, we can give an information-theoretic bound on the amount of information that has been leaked before the last encryption. But if there is no such bound, information theory cannot be used to derive such a bound. Instead, a computational assumption must be used. Yet, there seems to be no straightforward way to derive a useful statement (e.g., about the computational key leakage) that reaches across a polynomial number of instances from a single computational assumption using a hybrid argument. Of course, this excludes certain interactive assumptions, which essentially already assume security of the scheme in the first place. We do not believe that it is useful or interesting to investigate such constructions and assumptions.

In other words, we cannot use hybrid arguments since we do not know where to place the first hybrid step. This situation is similar (but not identical) to the case of selective decommitments [15] and adaptively secure encryption (e.g., [9]).

Hybrid (KEM/DEM) encryption schemes. Another common tool for constructing encryption schemes are hybrid encryption schemes (no relation to hybrid arguments). In a hybrid encryption scheme, a ciphertext consists of a KEM (key encapsulation mechanism) part and a DEM (data encapsulation mechanism) part. The KEM part of the ciphertext encapsulates a symmetric key K that is unrelated to the message M to be encrypted. The DEM part of the ciphertext is a (symmetric) encryption of M under K. The actual secret key of the hybrid scheme is the secret key that is needed to decrypt the KEM part. It is tempting to use a hybrid construction to get rid of the dependency of message and secret key. However, there still is a dependency between M and : the KEM ciphertext provides a relation between and K on the one hand, and the DEM ciphertext relates K and M on the other. Hybrid encryption techniques do not help to get rid of dependencies between message and secret key.

Similarly, hybrid encryption techniques cannot be used to increase the allowed message lengths of the scheme from the previous section. Concretely, it may be tempting to use the p-**BKDM** scheme as a KEM to encapsulate a short key K, and then to use that key K as secret key for a DEM which encrypts long messages with short keys. Unfortunately, this breaks the security proof of p-**BKDM** (and also, depending on the used DEM, also the security itself). Namely, the proof of p-**BKDM** depends not on the size of the KEM key K, but on the amount of released information about the actual KEM secret key (which corresponds to the length of the message in the KDM setting). So hybrid encryption does not help here, either.

Stateful KDM security. To nonetheless get a scheme that is secure in face of arbitrarily many encryptions of key-dependent messages, we propose encryption schemes. In a stateful encryption scheme, the secret key (i.e., the internal state) is updated on each encryption. (Decryption must then be synchronized with encryption: we assume that ciphertexts are decrypted in the order they got produced by encryption.) For such a stateful encryption scheme, there are essentially two interpretations of KDM security:

- the message may depend on the current secret key (i.e., state) only, or
- the message may depend on the current and all previously used secret keys (i.e., on the current and all previous states).

We call the first notion , and the second . Weak stateful KDM security can be thought of as KDM security in a setting in which erasures are trusted, and strong stateful KDM security mandates that erasures are trusted (in the most adversarial sense).

Definition 10 (Weak and strong stateful KDM security). Π weak stateful KDM security Π current Π strong stateful KDM security Π current and all previous states

Below we will give a scheme that circumvents the hybrid argument dilemma using precisely the fact that there is a changing state.

Relation to Black et al.'s notion of "stateful KDM security". Black et al. [7] already consider the potential KDM security of a stateful symmetric encryption scheme. They show that there . However, they showed this under the assumption that encryption is deterministic. In our definition, encryption is still probabilistic, even though stateful. We use the state update mechanism to using randomness, not instead of it. Their argument does not apply to our definition of stateful KDM security, neither to our weak nor to our strong variant.

Weak vs. strong stateful KDM security. For some applications, strong stateful KDM security is necessary: encrypting your hard drive (that may contain the secret key) cannot be done in a provably secure way with weak stateful KDM security. (Once the secret key gets to be processed by the scheme, the state may have already been updated, so that the message now depends on a [illegible] state.) Also, the notion of [illegible] (i.e., key K_i is encrypted under $K_{i+1 \bmod n}$) does not make sense with weak stateful KDM secure schemes. In these cases, the use of a [illegible] stateful KDM scheme is fine. However, it seems technically much more difficult to construct a strong stateful KDM secure scheme.

5.2 A Secure Scheme

We do not know how to fulfill strong stateful KDM security. (The issues that arise are similar as in the stateless case.) However, we [illegible] present a scheme that is secure in the sense of weak stateful KDM security.

Idea of the construction. Our scheme is a computational variant of p-**BKDM** (although its analysis will turn out to be very different). The main problem of p-**BKDM** is that the secret key runs out of entropy after too many KDM encryptions. Only as long as there is enough entropy left in K, a suitably independent random pad can be distilled for encryption. However, in a computational setting, randomness can be expanded with a pseudorandom generator, and some distilled, high-quality randomness can be used to generate more (pseudo-)randomness as a new key. More concretely, consider the following scheme:

Scheme 11 (The scheme sKDM (for "stateful KDM")). Let $\mathcal{UHF}$ be a family of universal hash functions that map $5k$-bit strings to k-bit strings, and let G be a pseudorandom generator (against uniform adversaries) that maps a k-bit seed to a $6k$-bit string. Define the stateful symmetric encryption scheme **sKDM** $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with security parameter $k \in \mathbb{N}$, message space $\{0,1\}^k$, and key space $\{0,1\}^{5k}$ through

- $\mathcal{K}(1^k)$ outputs a uniform random initial key (i.e., state) $K_0 \in \{0,1\}^{5k}$.
- $\mathcal{E}(1^k, K_j, M_j)$ proceeds as follows:
  1. sample $h_j \xleftarrow{\$} \mathcal{UHF}$,
  2. set $S_j := h_j(K_j)$,
  3. set $(K_{j+1}, P_j) := G(S)$,
  4. output $C_j := (h_j, P_j \oplus M_j)$.

  Ciphertext is C_j, and new key (i.e., state) is K_{j+1}.
- $\mathcal{D}(1^k, K_j, (h_j, D_j))$ proceeds as follows:
  1. set $S_j := h_j(K_j)$,
  2. set $(K_{j+1}, P_j) := G(S)$,
  3. output $M_j := P_j \oplus D_j$.

  Plaintext is M_j, and new key (i.e., state) is K_{j+1}.

Theorem 12. [illegible] G [illegible] **sKDM** [illegible]

Fix an adversary A that attacks **sKDM** in the sense of weak stateful KDM security. Say that, without loss of generality, A makes precisely $p(k)$ encryption queries for a positively-valued polynomial $p \in \mathbb{Z}[k]$. Assume that A has an advantage that is not negligible.

Preparation for hybrid argument. For $0 \leq j \leq p(k)$, define the hybrid game Game j as follows. Game j is the same as the weak stateful KDM game with adversary A, only that

- the first j encryption queries are answered as in the fake weak stateful KDM game (i.e., with encryptions of uniform and independent randomness), and
- the remaining queries are answered as in the real game (i.e., with encryptions of adversary-delivered functions evaluated at the secret key).

Base step for hybrid argument. We will reduce distinguishing between two adjacent games to some computational assumption. We will now first formulate this assumption. Let $K \in \{0,1\}^{5k}$ be uniformly distributed, and let $M \in \{0,1\}^k$ be arbitrary (in particular, M can be a function of K). Then by Lemma 1 it follows that $\delta(M, h, h(K) \ ; \ M, h, U_k) \leq 2^{-k}$ for independently sampled $h \xleftarrow{\$} \mathcal{UHF}$ and independent uniform $U_k \in \{0,1\}^k$. (Actually, in this case we could even use the original version of the Leftover Hash Lemma [18].) This implies

$$\delta(M, h, G(h(K)) \ ; \ M, h, G(U_k)) \leq 2^{-k},$$

from which the computational indistinguishability chain

$$\underbrace{(M, h, G(h(K)))}_{=:D^R} \approx (M, h, G(U)) \approx \underbrace{(M, h, U_{6k})}_{=:D^F} \qquad (4)$$

for independent uniform $U_{6k} \in \{0,1\}^{6k}$ follows by assumption on G. For our hybrid argument, it is important that (4) even holds when M is a function of K chosen by the distinguisher.

Hybrid argument. We will now construct from adversary A an adversary B that contradicts (4) by distinguishing D^R and D^F. This contradiction then concludes our proof. Let n denote the number of keys. Let μ_i denote the index of the key chosen by A for the i-th encryption. Let g_i denote the function chosen by A in the i-th encryption. Then, the adversary B chooses some $j \in \{1, \ldots, p(k)\}$ uniformly at random and then performs the following simulation for A:

– The first $j-1$ encryptions requested by A are simulated as fake encryptions (i.e., with random messages). This is possible without using the keys since for a random message, $h_i(K_{\mu_i})$ is information-theoretically hidden in the ciphertext.

– For the j-th encryption, B chooses K_μ randomly for all $\mu \neq \mu_j$ and defines[4] $M(K) := g_j(K_1, \ldots, K_{\mu_j - 1}, K, K_{\mu_j+1}, \ldots, K_n)$ and requests an input $D =:$

[4] Note that in this *function definition*, K is the argument while the K_{μ_i} are hardwired. In particular, B does not need to know the actual value of K for this step.

$(M, h, (P, K'))$ with that M. (Note that D may be D^R or D^F.) Then B sets the new key $K_{\mu_j} := K'$ and gives $(h, M \oplus P)$ as the ciphertext to A.
- For all further encryptions queries, B computes the real ciphertext using the keys $K_1, \ldots, K_n$ produced in the preceding steps.
- Finally, B outputs the output of A.

It is now easy to verify that if B gets D^R as input, B simulates the Game $j-1$, and if B gets D^F as input, B simulates the Game j. Hence

$$\begin{aligned}
&\Pr\left[B(D^R) = 1\right] - \Pr\left[B(D^F) = 1\right] \\
&\quad = \frac{1}{p(k)} \sum_{j=1}^{p(k)} \Pr\left[A = 1 \text{ in Game } j-1\right] - \frac{1}{p(k)} \sum_{j=1}^{p(k)} \Pr\left[A = 1 \text{ in Game } j\right] \\
&\quad = \frac{1}{p(k)} \big(\Pr\left[A = 1 \text{ in Game } 0\right] - \Pr\left[A = 1 \text{ in Game } p(k)\right]\big).
\end{aligned}$$

The right hand side is not negligible by assumption, thus the right hand side is not negligible either. This contradicts (4) and thus concludes the proof.

5.3 The Usefulness of Stateful KDM Security

In a sense, strong stateful KDM security is "just as good" as standard KDM security. Arbitrarily large messages (in particular keys) can be encrypted by splitting up the message into parts and encrypting each part individually. The key-depencies of the message parts can be preserved, since the dependencies across states (i.e., dependencies on earlier keys) are allowed. This technique is generally possible with weak stateful KDM security. We know of no weakly stateful KDM secure scheme with which one could securely encrypt one's own key (let alone construct key cycles).

But despite the drawbacks of weak stateful KDM security, we believe that this notion is still useful: first, it serves as a stepping stone towards achieving strong stateful KDM security (or even stateless KDM security). Second, in certain applications, weak stateful KDM security might be sufficient. Imagine, e.g., a setting in which the encrypted message contains side-channel information (like, say, internal measurements from the encryption device) on the internal state/secret key. If we assume that the old state is erased after encryption, the side-channel information only refers to the internal state, and weak stateful KDM security is enough to provide message secrecy. Third, weak stateful KDM security provides an alternative assumption to the assumption of absence of key cycles in the formal protocol analysis setting. Instead of assuming the absence of key cycles (this assumption may not make sense in a scheme in which the key space is larger than the message space), we can assume that the encrypted terms depend only on the current internal state of the encryption algorithm. This assumption is still a strengthening of standard IND-CPA security and makes sense, since the encryption algorithm is only used to encrypt.

References

1. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology 15(2), 103–127 (2002)
2. Abão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness of formal encryption in the presence of key-cycles. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 374–396. Springer, Heidelberg (2005)
3. Backes, M., Pfitzmann, B., Scedrov, A.: Key-dependent message security under active attacks – BRSIM/UC-soundness of symbolic encryption with key cycles. In: Proceedings of CSF 2007, pp. 112–124. IEEE Computer Society Press, Los Alamitos (2007)
4. Backes, M., Pfitzmann, B., Waidner, M.: A composable cryptographic library with nested operations. In: Proceedings of CCS 2003, pp. 220–230. ACM Press, New York (2003)
5. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings of FOCS 1997, pp. 394–403. IEEE Computer Society Press, Los Alamitos (1997)
6. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
7. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)
8. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
9. Canetti, R., Halevi, S., Katz, J.: Adaptively-secure, non-interactive public-key encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 150–168. Springer, Heidelberg (2005)
10. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
11. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
12. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
13. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: Proceedings of STOC 1991, pp. 542–552. ACM Press, New York (1991)
14. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–208 (1983)
15. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.: Magic functions. Journal of the ACM 50(6), 852–921 (2003)
16. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
17. Halevi, S., Krawczyk, H.: Security under key-dependent inputs. In: Proceedings of CCS 2007, ACM Press, New York (2007)

18. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
19. Hofheinz, D., Müller-Quade, J., Steinwandt, R.: On modeling IND-CCA security in cryptographic protocols. Tatra Mountains Mathematical Publications 33, 83–97 (2006)
20. Micciancio, D., Warinschi, B.: Soundness of formal encryption in the presence of active adversaries. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 133–151. Springer, Heidelberg (2004)
21. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)

The Twin Diffie-Hellman Problem and Applications

David Cash*, Eike Kiltz**, and Victor Shoup***

[1] College of Computing, Georgia Institute of Technology, USA
cdc@gatech.edu
[2] Cryptology & Information Security Group, CWI Amsterdam, The Netherlands
kiltz@cwi.nl
[3] Dept. of Computer Science, New York University, Courant Institute, 251 Mercer Street, New York, NY 10012, USA
shoup@cs.nyu.edu

Abstract. We propose a new computational problem called the *twin Diffie-Hellman problem*. This problem is closely related to the usual (computational) Diffie-Hellman problem and can be used in many of the same cryptographic constructions that are based on the Diffie-Hellman problem. Moreover, the twin Diffie-Hellman problem is at least as hard as the ordinary Diffie-Hellman problem. However, we are able to show that the twin Diffie-Hellman problem remains hard, even in the presence of a decision oracle that recognizes solutions to the problem — this is a feature not enjoyed by the ordinary Diffie-Hellman problem. In particular, we show how to build a certain "trapdoor test" which allows us to effectively answer such decision oracle queries, without knowing any of the corresponding discrete logarithms. Our new techniques have many applications. As one such application, we present a new variant of ElGamal encryption with very short ciphertexts, and with a very simple and tight security proof, in the random oracle model, under the assumption that the ordinary Diffie-Hellman problem is hard. We present several other applications as well, including: a new variant of Diffie and Hellman's non-interactive key exchange protocol; a new variant of Cramer-Shoup encryption, with a very simple proof in the standard model; a new variant of Boneh-Franklin identity-based encryption, with very short ciphertexts; a more robust version of a password-authenticated key exchange protocol of Abdalla and Pointcheval.

1 Introduction

In some situations, basing security proofs on the hardness of the Diffie-Hellman problem is hindered by the fact that recognizing correct solutions is also apparently hard (indeed, the hardness of the latter problem is the Decisional Diffie-Hellman assumption). There are a number of ways for circumventing these technical difficulties. One way is to simply make a stronger assumption, namely,

* Part of this work completed while at CWI.
** Supported by the research program Sentinels.
*** Supported by NSF award number CNS-0716690.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 127–145, 2008.

that the Diffie-Hellman problem remains hard, even given access to a corresponding decision oracle. Another way is to work with groups that are equipped with efficient pairings, so that such a decision oracle is immediately available. However, we would like to avoid making stronger assumptions, or working with specialized groups, if at all possible.

In this paper, we introduce a new problem, the [illegible], which has the following interesting properties:

- the twin Diffie-Hellman problem can easily be employed in many cryptographic constructions where one would usually use the ordinary Diffie-Hellman problem, without imposing a terrible efficiency penalty;
- the twin Diffie-Hellman problem is hard, even given access to a corresponding decision oracle, assuming the ordinary Diffie-Hellman problem (without access to any oracles) is hard.

Using the twin Diffie-Hellman problem, we construct a new variant of ElGamal encryption that is secure against chosen ciphertext attack, in the random oracle model, under the assumption that the ordinary Diffie-Hellman problem is hard. Compared to other ElGamal variants with similar security properties, our scheme is attractive in that it has very short ciphertexts, and a very simple and tighter security proof.

At the heart of our method is a "trapdoor test" that allows us to implement an effective decision oracle for the twin Diffie-Hellman problem, without knowing any of the corresponding discrete logarithms. This trapdoor test has many applications, including: a new variant of Diffie and Hellman's non-interactive key exchange protocol [10], which is secure in the random oracle model assuming the Diffie-Hellman problem is hard; a new variant of Cramer-Shoup encryption [8] with a very simple security proof, in the standard model, under the [illegible] decisional Diffie-Hellman assumption; a new variant of Boneh-Franklin identity-based encryption [5], with very short ciphertexts, and a simple and tighter security proof in the random oracle model, assuming the bilinear Diffie-Hellman problem is hard; a very simple and efficient method of securing a password-authenticated key exchange protocol of Abdalla and Pointcheval [2] against server compromise, which can be proved secure, using our trapdoor test, in the random oracle model, under the Diffie-Hellman assumption.

1.1 Hashed ElGamal Encryption and Its Relation to the Diffie-Hellman Problem

To motivate the discussion, consider the "hashed" ElGamal encryption scheme [1]. This public-key encryption scheme makes use of a group $\mathbb{G}$ of prime order q with generator $g \in \mathbb{G}$, a hash function H, and a symmetric cipher (E, D). A public key for this scheme is a random group element X, with corresponding secret key x, where $X = g^x$. To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes

$$Y := g^y, \quad Z := X^y, \quad k := \mathsf{H}(Y, Z), \quad c := \mathsf{E}_k(m),$$

and the ciphertext is (Y, c). Decryption works in the obvious way: given the ciphertext (Y, c), and secret key x, one computes

$$Z := Y^x, \quad k := \mathsf{H}(Y, Z), \quad m := \mathsf{D}_k(c).$$

THE DIFFIE-HELLMAN ASSUMPTION. Clearly, the hashed ElGamal encryption scheme is secure [illegible] it is hard to compute Z, given the values X and Y. Define

$$\mathrm{dh}(X, Y) := Z, \quad \text{where } X = g^x,\ Y = g^y, \text{ and } Z = g^{xy}. \tag{1}$$

The problem of computing $\mathrm{dh}(X, Y)$ given random $X, Y \in \mathbb{G}$ is the [illegible]. The [illegible] asserts that this problem is hard. However, this assumption is [illegible] sufficient to establish the security of hashed ElGamal against a [illegible], regardless of what security properties the hash function H may enjoy.

To illustrate the problem, suppose that an adversary selects group elements $\hat{Y}$ and $\hat{Z}$ in some arbitrary way, and computes $\hat{k} := \mathsf{H}(\hat{Y}, \hat{Z})$ and $\hat{c} := \mathsf{E}_{\hat{k}}(\hat{m})$ for some arbitrary message $\hat{m}$. Further, suppose the adversary gives the ciphertext $(\hat{Y}, \hat{c})$ to a "decryption oracle," obtaining the decryption m. Now, it is very likely that $\hat{m} = m$ if and only if $\hat{Z} = \mathrm{dh}(X, \hat{Y})$. Thus, the decryption oracle can be used by the adversary as an oracle to answer questions of the form "is $\mathrm{dh}(X, \hat{Y}) = \hat{Z}$?" for group elements $\hat{Y}$ and $\hat{Z}$ of the adversary's choosing. In general, the adversary would not be able to efficiently answer such questions on his own, and so the decryption oracle is leaking some information about that secret key x which could conceivably be used to break the encryption scheme.

THE STRONG DH ASSUMPTION. Therefore, to establish the security of hashed ElGamal against chosen ciphertext attack, we need a stronger assumption. For $X, \hat{Y}, \hat{Z} \in \mathbb{G}$, define the predicate

$$\mathrm{dhp}(X, \hat{Y}, \hat{Z}) := \quad \mathrm{dh}(X, \hat{Y}) \stackrel{?}{=} \hat{Z}.$$

At a bare minimum, we need to assume that it is hard to compute $\mathrm{dh}(X, Y)$, given random $X, Y \in \mathbb{G}$, along with access to a [illegible] for the predicate $\mathrm{dhp}(X, \cdot, \cdot)$, which on input $(\hat{Y}, \hat{Z})$, returns $\mathrm{dhp}(X, \hat{Y}, \hat{Z})$. This assumption is called the [illegible] [1].[1] Moreover, it is not hard to prove, if H is modeled as a random oracle, that hashed ElGamal is secure against chosen ciphertext attack under the strong DH assumption, and under the assumption that the underlying symmetric cipher is itself secure against chosen ciphertext attack. This was proved in [1,21], for a variant scheme in which Y is not included in the hash; including Y in the hash gives a more efficient security reduction (see [9]). Note that the strong DH assumption is different (and weaker) than the so called [illegible] [24] where an adversary gets access to a [illegible] decision oracle for the predicate $\mathrm{dhp}(\cdot, \cdot, \cdot)$, which on input $(\hat{X}, \hat{Y}, \hat{Z})$, returns $\mathrm{dhp}(\hat{X}, \hat{Y}, \hat{Z})$.

[1] We remark that in more recent papers the name strong DH assumption also sometimes refers to a different assumption defined over bilinear maps [3]. We follow the original terminology from [1].

1.2 The Twin Diffie-Hellman Assumptions

For general groups, the strong DH assumption may be strictly stronger than the DH assumption. One of the main results of this paper is to present a slightly modified version of the DH problem that is just as useful as the (ordinary) DH problem, and which is just as hard as the (ordinary) DH problem, [illegible]. Using this, we get a modified version of hashed ElGamal encryption which can be proved secure under the (ordinary) DH assumption, in the random oracle model. This modified system is just a bit less efficient than the original system.

Again, let $\mathbb{G}$ be a cyclic group with generator g, and of prime order q. Let dh be defined as in (1). Define the function

$$\begin{aligned} 2\mathrm{dh}: \quad & \mathbb{G}^3 \to \mathbb{G}^2 \\ & (X_1, X_2, Y) \mapsto (\mathrm{dh}(X_1, Y), \mathrm{dh}(X_2, Y)). \end{aligned}$$

We call this the [illegible]. One can also define a corresponding [illegible]:

$$2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) := \quad 2\mathrm{dh}(X_1, X_2, \hat{Y}) \stackrel{?}{=} (\hat{Z}_1, \hat{Z}_2).$$

The [illegible] states it is hard to compute $2\mathrm{dh}(X_1, X_2, Y)$, given random $X_1, X_2, Y \in \mathbb{G}$. It is clear that the DH assumption implies the twin DH assumption. The [illegible] states that it is hard to compute $2\mathrm{dh}(X_1, X_2, Y)$, given random $X_1, X_2, Y \in \mathbb{G}$, along with access to a [illegible] for the predicate $2\mathrm{dhp}(X_1, X_2, \cdot, \cdot, \cdot)$, which on input $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$, returns $2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$.

One of our main results is the following:

Theorem 1. [illegible]

The non-trivial direction to prove is that the DH assumption implies the strong twin DH assumption.

A Trapdoor Test. While Theorem 1 has direct applications, the basic tool that is used to prove the theorem, which is a kind of "trapdoor test," has even wider applications. Roughly stated, the trapdoor test works as follows: given a random group element X_1, we can efficiently construct a random group element X_2, together with a secret "trapdoor" τ, such that

- X_1 and X_2 are independent (as random variables), and
- if we are given group elements $\hat{Y}, \hat{Z}_1, \hat{Z}_2$, computed as functions of X_1 and X_2 (but not τ), then using τ, we can efficiently evaluate the predicate $2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$, making a mistake with only negligible probability.

We note that our trapdoor test actually appears implicitly in Shoup's DH self-corrector [28]; apparently, its implications were not understood at the time, although the techniques of Cramer and Shoup [8] are in some sense an extension of the idea. Due to space constraints we must defer the details of the connection between our trapdoor test and Shoup's DH self-corrector to the full version of this paper.

1.3 Applications and Results

The twin ElGamal encryption scheme. Theorem 1 suggests the following [illegible]. This scheme makes use of a hash function H and a symmetric cipher (E, D). A public key for this scheme is a pair of random group elements (X_1, X_2), with corresponding secret key is (x_1, x_2), where $X_i = g^{x_i}$ for $i = 1, 2$. To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes

$$Y := g^y, \quad Z_1 := X_1^y, \quad Z_2 := X_2^y, \quad k := \mathsf{H}(Y, Z_1, Z_2), \quad c := \mathsf{E}_k(m),$$

and the ciphertext is (Y, c). Decryption works in the obvious way: given the ciphertext (Y, c), and secret key (x_1, x_2), one computes

$$Z_1 := Y^{x_1}, \quad Z_2 := Y^{x_2}, \quad k := \mathsf{H}(Y, Z_1, Z_2), \quad m := \mathsf{D}_k(c).$$

The arguments in [1] and [9] trivially carry over, so that one can easily show that the twin ElGamal encryption scheme is secure against chosen ciphertext attack, under the strong twin DH assumption, and under the assumption that (E, D) is secure against chosen ciphertext attack, if H is modeled as a random oracle. Again, by Theorem 1, the same holds under the (ordinary) DH assumption.

Note that the ciphertexts for this scheme are extremely compact — no redundancy is added, as in the Fujisaki-Okamoto transformation [11]. Moreover, the security reduction for our scheme is very tight. We remark that this seems to be the first DH-based encryption scheme with short ciphertexts. All other known constructions either add redundancy to the ciphertext [11,25,29,7,18] or resort to assumptions stronger than DH [1,9,21].

The twin DH key-exchange protocol. In their paper [10], Diffie and Hellman presented the following simple, [illegible] key exchange protocol. Alice chooses a random $x \in \mathbb{Z}_q$, computes $X := g^x \in \mathbb{G}$, and publishes the pair $(\texttt{Alice}, X)$ is a public directory. Similarly, Bob chooses a random $y \in \mathbb{Z}_q$, computes $Y := g^y \in \mathbb{G}$, and publishes the pair $(\texttt{Bob}, Y)$ in a public directory. Alice and Bob may compute the shared value $Z := g^{xy} \in \mathbb{G}$, as follows: Alice retrieves Bob's entry from the directory and computes Z as Y^x, while Bob retrieves Alice's key X, and computes Z as X^y. Before using the value Z, it is generally a good idea to hash it, together with Alice's and Bob's identities, using a cryptographic hash function H. Thus, the key that Alice and Bob actually use to encrypt data using a symmetric cipher is $k := \mathsf{H}(\texttt{Alice}, \texttt{Bob}, Z)$.

Unfortunately, the status of the security of this scheme is essentially the same as that of the security of hashed ElGamal against chosen ciphertext attack, if we allow an adversary to place arbitrary public keys in the public directory (without requiring some sort of "proof of possession" of a secret key).

To avoid this problem, we define the [illegible], as follows: Alice's public key is (X_1, X_2), and her secret key is (x_1, x_2), where $X_i = g^{x_i}$ for $i = 1, 2$; similarly, Bob's public key is (Y_1, Y_2), and his secret key is (y_1, y_2), where $Y_i = g^{y_i}$ for $i = 1, 2$; their shared key is

$$k := \mathsf{H}(\texttt{Alice}, \texttt{Bob}, \mathrm{dh}(X_1, Y_1), \mathrm{dh}(X_1, Y_2), \mathrm{dh}(X_2, Y_1), \mathrm{dh}(X_2, Y_2)),$$

where H is a hash function. Of course, Alice computes the 4-tuple of group elements in the hash as

$$(Y_1^{x_1}, Y_2^{x_1}, Y_1^{x_2}, Y_2^{x_2}),$$

and Bob computes them as

$$(X_1^{y_1}, X_1^{y_2}, X_2^{y_1}, X_2^{y_2}).$$

Using the "trapdoor test," it is a simple matter to show that the twin DH protocol satisfies a natural and strong definition of security, under the (ordinary) DH assumption, if H is modeled as a random oracle.

A variant of Cramer-Shoup encryption. We present a variant of the public-key encryption scheme by Cramer and Shoup [8]. Using our trapdoor test, along with techniques originally developed for identity-based encryption [3], we give an extremely simple proof of its security against chosen-ciphertext attack, in the standard model, under the Decisional DH assumption [12]: given X and Y, it is hard to distinguish $\mathrm{dh}(X, Y)$ from Z, for random $X, Y, Z \in \mathbb{G}$. In fact, our proof works under the weaker [illegible] Decisional DH assumption: given X and Y, it is hard to distinguish $\mathsf{H}(\mathrm{dh}(X, Y))$ from k, for random $X, Y \in \mathbb{G}$, and random k in the range of H. Note that the original Cramer-Shoup scheme cannot be proved secure under this weaker assumption — their security relies in an essential way on the Decisional DH assumption.

As a simple extension of this idea, we can obtain a new analysis of a scheme given in [17]. There, a variant of the Kurosawa-Desmedt encryption scheme is given and proved secure under the decisional DH assumption. Our analysis provides further theoretical understanding. Due to space constraints we must defer the details of this construction to the full version of this paper.

Obviously, our variants are secure under the DH assumption if H is modeled as a random oracle. We also note that by using the Goldreich-Levin theorem, a simple extension of our scheme, which is still fairly practical, can be proved secure against chosen ciphertext attack under the DH assumption.

The observation that a variant of the Cramer-Shoup encryption scheme can be proved secure under the hashed Decisional DH assumption was also made by Brent Waters, in unpublished work (personal communication, 2006) and independently by Goichiro Hanaoka and Kaoru Kurosawa, also in unpublished work [16].

Identity-based encryption. Strong versions of assumptions also seem necessary to analyze some identity-based encryption (IBE) schemes that use bilinear pairings. As a further contribution, we give a twin version of the [illegible] DH (BDH) assumption and prove that the (interactive) strong twin BDH assumption is implied by the standard BDH assumption.

The well-known IBE scheme of Boneh and Franklin [5] achieves security against chosen ciphertext, in the random oracle model, by applying the Fujisaki-Okamoto transformation. Our techniques give a different scheme with shorter ciphertexts, and a tighter security reduction. The same technique can also be applied to the scheme by Kasahara and Sakai [27] which is based on a stronger bilinear assumption but has improved efficiency.

Other applications. Our twinning technique and in particular the trapdoor test can be viewed as a general framework that allows to "update" a protocol Π whose security relies on the strong DH assumption to a protocol Π' that has roughly the same complexity as Π, but whose security is solely based on the DH assumption. Apart from the applications mentioned above, we remark that this technique can also be applied to the undeniable signatures and designated confirmer signatures from [24] and the key-exchange protocols from [19].

As another application of our trapdoor test, one can easily convert the very elegant and efficient protocol of Abdalla and Pointcheval [2] for password-authenticated key exchange, into a protocol that provides security against server compromise, without adding any messages to the protocol, and still basing the security proof, in the random oracle model, on the DH assumption. For lack of space, this application will be further discussed in the full version.

2 A Trapdoor Test and a Proof of Theorem 1

It is not hard to see that the strong twin DH implies the DH assumption. To prove that the DH implies the strong twin DH assumption, we first need our basic tool, a "trapdoor test". Its purpose will be intuitively clear in the proof of Theorem 1: in order to reduce the strong twin DH assumption to the DH assumption, the DH adversary will have to answer decision oracle queries without knowing the discrete logarithms of the elements of the strong twin DH problem instance. This tool gives us a method for doing so.

Theorem 2 (Trapdoor Test). [illegible] $\mathbb{G}$ [illegible] q [illegible] $g \in \mathbb{G}$ [illegible] X_1, r, s [illegible] X_1 [illegible] $\mathbb{G}$ [illegible] r, s [illegible] $\mathbb{Z}_q$ [illegible] $X_2 := g^s/X_1^r$ [illegible] $\hat{Y}, \hat{Z}_1, \hat{Z}_2$ [illegible] $\mathbb{G}$ [illegible] X_1 [illegible] X_2 [illegible]

- [illegible] X_2 [illegible] $\mathbb{G}$
- [illegible] X_1 [illegible] X_2 [illegible]
- [illegible] $X_1 = g^{x_1}$ [illegible] $X_2 = g^{x_2}$ [illegible]

$$\hat{Z}_1^r \hat{Z}_2 = \hat{Y}^s \tag{2}$$

[illegible]

$$\hat{Z}_1 = \hat{Y}^{x_1} \wedge \hat{Z}_2 = \hat{Y}^{x_2} \tag{3}$$

[illegible] $1/q$ [illegible] (2) [illegible] (3) [illegible]

Observe that $s = rx_1 + x_2$. It is easy to verify that X_2 is uniformly distributed over $\mathbb{G}$, and that X_1, X_2, r are mutually independent, from which (i) and (ii) follow. To prove (iii), condition on fixed values of X_1 and X_2. In the resulting conditional probability space, r is uniformly distributed over $\mathbb{Z}_q$, while x_1, x_2, $\hat{Y}$, $\hat{Z}_1$, and $\hat{Z}_2$ are fixed. If (3) holds, then by multiplying together the two equations in (3), we see that (2) certainly holds. Conversely, if (3) does not hold, we show that (2) holds with probability at most $1/q$. Observe that (2) is equivalent to

$$(\hat{Z}_1/\hat{Y}^{x_1})^r = \hat{Y}^{x_2}/\hat{Z}_2. \tag{4}$$

It is not hard to see that if $\hat{Z}_1 = \hat{Y}^{x_1}$ and $\hat{Z}_2 \neq \hat{Y}^{x_2}$, then (4) certainly does not hold. This leaves us with the case $\hat{Z}_1 \neq \hat{Y}^{x_1}$. But in this case, the left hand side of (4) is a random element of $\mathbb{G}$ (since r is uniformly distributed over $\mathbb{Z}_q$), but the right hand side is a fixed element of $\mathbb{G}$. Thus, (4) holds with probability $1/q$ in this case.

Using this tool, we can easily prove Theorem 1. So that we can give a concrete security result, let us define some terms. For an adversary $\mathcal{B}$, let us define his , denoted $\mathsf{AdvDH}_{\mathcal{B},\mathbb{G}}$, to be the probability that $\mathcal{B}$ computes $\mathrm{dh}(X, Y)$, given random $X, Y \in \mathbb{G}$. For an adversary $\mathcal{A}$, let us define his , denoted $\mathsf{Adv2DH}_{\mathcal{A},\mathbb{G}}$, to be the probability that $\mathcal{A}$ computes $\mathrm{2dh}(X_1, X_2, Y)$, given random $X_1, X_2, Y \in \mathbb{G}$, along with access to a for the predicate $\mathrm{2dhp}(X_1, X_2, \cdot, \cdot, \cdot)$, which on input $\hat{Y}, \hat{Z}_1, \hat{Z}_2$, returns $\mathrm{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$.

Theorem 1 is a special case of the following:

Theorem 3. $\mathcal{A}$ Q_{d} τ $\mathcal{B}$ $\mathcal{B}$ τ $O(Q_{\mathrm{d}} \log q)$

$$\mathsf{Adv2DH}_{\mathcal{A},\mathbb{G}} \leq \mathsf{AdvDH}_{\mathcal{B},\mathbb{G}} + Q_{\mathrm{d}}/q.$$

$\mathcal{B}$ ” $1 - 1/q$

Our DH adversary $\mathcal{B}$ works as follows, given a challenge instance (X, Y) of the DH problem. First, $\mathcal{B}$ chooses $r, s \in \mathbb{Z}_q$ at random, sets $X_1 := X$ and $X_2 := g^s/X_1^r$, and gives $\mathcal{A}$ the challenge instance (X_1, X_2, Y). Second, $\mathcal{B}$ processes each decision query $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ by testing if $\hat{Z}_1\hat{Z}_2^r = \hat{Y}^s$ holds. Finally, if and when $\mathcal{A}$ outputs (Z_1, Z_2), $\mathcal{B}$ tests if this output is correct by testing if $Z_1 Z_2^r = Y^s$ holds; if this does not hold, then $\mathcal{B}$ outputs “failure,” and otherwise, $\mathcal{B}$ outputs Z_1. The proof is easily completed using Theorem 2.

3 Twin ElGamal Encryption

3.1 Model and Security

We recall the definition for chosen ciphertext security of a public-key encryption scheme, denoted PKE. Consider the usual chosen ciphertext attack game, played between a challenger and a adversary $\mathcal{A}$:

1. The challenger generates a public key/secret key pair, and gives the public key to $\mathcal{A}$;
2. $\mathcal{A}$ makes a number of to the challenger; each such query is a ciphertext $\hat{C}$; the challenger decrypts $\hat{C}$, and sends the result to $\mathcal{A}$;
3. $\mathcal{A}$ makes one , which is a pair of messages (m_0, m_1); the challenger chooses $b \in \{0,1\}$ at random, encrypts m_b, and sends the resulting ciphertext C to $\mathcal{A}$;
4. $\mathcal{A}$ makes more , just as in step 2, but with the restriction that $\hat{C} \neq C$;
5. $\mathcal{A}$ outputs $\hat{b} \in \{0,1\}$.

The advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}}$ is defined to be $|\Pr[\hat{b} = b] - 1/2|$. The scheme PKE is said to be secure against chosen ciphertext attack if for all efficient adversaries $\mathcal{A}$, the advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}}$ is negligible.

If we wish to analyze a scheme PKE in the random oracle model, then hash functions are replaced by random oracle queries as appropriate, and both challenger and adversary are given access to the random oracle in the above attack game. We write $\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{PKE}}$ for the corresponding advantage in the random oracle model.

If $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ is a symmetric cipher, then one defines security against chosen ciphertext attack in exactly the same way, except that in step 1 of the above attack game, the challenger simply generates a secret key and step 2 of the above attack game is left out. The advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{SE}}$ is defined in exactly the same way, and SE is said to be secure against chosen ciphertext attack if for all efficient adversaries $\mathcal{A}$, the advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{SE}}$ is negligible.

The usual construction of a chosen-ciphertext secure symmetric encryption scheme is to combine a one-time pad and a message-authentication code (MAC). We remark that such schemes do not necessarily add any redundancy to the symmetric ciphertext. In fact, Phan and Pointcheval [26] showed that a [13] directly implies a length-preserving chosen-ciphertext secure symmetric encryption scheme that avoids the usual overhead due to the MAC. In practice one can use certain modes of operation (e.g., CMC [15]) to encrypt large messages. The resulting scheme is chosen-ciphertext secure provided that the underlying block-cipher is a strong PRP.

3.2 Security of the Twin ElGamal Scheme

We are now able to establish the security of the twin ElGamal encryption scheme described in §1.3, which we denote $\mathsf{PKE}_{2\mathrm{dh}}$. The security will be based on the strong twin DH assumption, of course, and this allows us to borrow the "oracle patching" technique from previous analyses of hashed ElGamal encryption based on the strong DH assumption. We stress, however, that unlike previous applications of this technique, the end result is a scheme based on the original DH assumption.

Theorem 4. [illegible] H [illegible] $\mathsf{PKE}_{2\mathrm{dh}}$ [illegible] $\mathcal{A}$ [illegible] $\mathsf{PKE}_{2\mathrm{dh}}$ [illegible] $\mathcal{A}$ [illegible] τ [illegible] Q_{h} [illegible] Q_{d} [illegible] $\mathcal{B}_{\mathrm{dh}}$ [illegible] $\mathcal{B}_{\mathrm{sym}}$ [illegible] SE [illegible] $\mathcal{B}_{\mathrm{dh}}$ [illegible] $\mathcal{B}_{\mathrm{sym}}$ [illegible] τ [illegible] $O((Q_{\mathrm{h}} + Q_{\mathrm{d}}) \log q)$ [illegible]

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{PKE}_{2\mathrm{dh}}} \leq \mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh}},\mathbb{G}} + \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}} + Q_{\mathrm{h}}/q.$$

Given the equivalence between the strong 2DH and the DH assumption from Theorem 1, the proof of Theorem 4 is quite standard, but must be deferred to the full version.

Instantiating $\mathsf{PKE}_{2\mathrm{dh}}$ with a length-preserving chosen-ciphertext secure symmetric encryption scheme, we obtain a DH-based chosen-ciphertext secure encryption scheme with the following properties.

Optimal ciphertext overhead. The ciphertext overhead, i.e. ciphertext size minus plaintext size, is exactly one group element, which is optimal for Diffie-Hellman based schemes. For concreteness, for $\kappa = 128$ bit security, a typical implementation in elliptic curve groups gives a concrete ciphertext overhead of 256 bits.

Encryption/decryption efficiency. Encryption needs three exponentiations in $\mathbb{G}$, one of which is to the fixed-base g (that can be shared among many public-keys). Decryption only needs one sequential exponentiation in $\mathbb{G}$ to compute Y^{x_1} and Y^{x_2} simultaneously, which is nearly as efficient as one single exponentiation (see, e.g., [23]).

4 Non-interactive Key Exchange

In this section we give a model and security definition for non-interactive key exchange and then analyze the twin DH protocol from section §1.3. Strangely, after the seminal work of Diffie and Hellman on this subject, it does not seem to have been explored further in the literature, except in the identity-based setting.

4.1 Model and Security

A non-interactive key exchange scheme KE consists of two algorithms: one for key generation and one for computing paired keys. The key generation algorithm is probabilistic and outputs a public key/secret key pair. The paired key algorithm takes as input an identity and public key along with another identity and a secret key, and outputs a shared key for the two identities. Here, identities are arbitrary strings chosen by the users, and the key authority does not generate keys itself but acts only as a phonebook.

For security we define an experiment between a challenger and an adversary $\mathcal{A}$. In this experiment, the challenger takes a random bit b as input and answers oracle queries for $\mathcal{A}$ until $\mathcal{A}$ outputs a bit $\hat{b}$. The challenger answers the following types of queries for $\mathcal{A}$:

Register honest user ID. $\mathcal{A}$ supplies a string ID. The challenger runs the key generation algorithm to generate a public key/secret key pair (pk, sk) and records the tuple (honest, ID, pk, sk) for later. The challenger returns pk to $\mathcal{A}$.

Register corrupt user ID. In this type of query, $\mathcal{A}$ supplies both the string ID and a public key pk. The challenger records the tuple (corrupt, ID, pk) for later.

Get honest paired key. Here $\mathcal{A}$ supplies two identities ID, ID' that were registered as honest users. Now the challenger uses the bit b: if $b = 0$, the challenger runs the paired key algorithm using the public key for ID and the secret key for ID'. If $b = 1$, the challenger generates a random key, records it for later, and returns that to the adversary. To keep things consistent, the challenger returns the same random key for the set $\{ID, ID'\}$ every time $\mathcal{A}$ queries for their paired key (perhaps in reversed order).

Get corrupt paired key. Here $\mathcal{A}$ supplies two identities ID, ID', where ID was registered as corrupt and ID' was registered as honest. The challenger runs the paired key algorithm using the public key for ID and the secret key for ID' and returns the paired key.

When the adversary finally outputs $\hat{b}$, it wins the experiment if $\hat{b} = b$. For an adversary $\mathcal{A}$, we define its advantage $\mathsf{AdvKA}_{\mathcal{A},\mathsf{KE}}$ in this experiment to be $|\Pr[\hat{b} = b] - 1/2|$. When a hash function is modeled as a random oracle in the experiment, we denote the adversary's advantage by $\mathsf{AdvKA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{KE}}$. We say that a non-interactive key-exchange scheme KE is [illegible] if for all efficient adversaries $\mathcal{A}$, the advantage $\mathsf{AdvKA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{KE}}$ is negligible.

We note that in the ideal version of the experiment above (when $b = 1$), the challenger returns the same random key for the honest paired key queries for (ID, ID') and (ID', ID). This essentially means that there should be no concept of "roles" in the model and that protocols should implement something like a canonical ordering of all the identities to implicitly define roles if needed.

4.2 Security of the Twin DH Protocol

As stated above, we can prove the twin DH protocol secure under the DH assumption using our trapdoor test. We denote the twin DH protocol by $\mathsf{KA}_{2\mathrm{dh}}$. A complete proof will be given in the full version.

Theorem 5. [illegible] H [illegible] $\mathsf{KA}_{2\mathrm{dh}}$ [illegible]

[illegible] $\mathcal{A}$ [illegible] $\mathsf{KA}_{2\mathrm{dh}}$ [illegible] $\mathcal{A}$ [illegible] τ [illegible] Q [illegible] $\mathcal{B}_{\mathrm{dh}}$ [illegible] τ [illegible] $O(Q \log q)$ [illegible]

$$\mathsf{AdvKA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{KA}_{2\mathrm{dh}}} \leq 2\mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh}},\mathbb{G}} + 4Q/q.$$

5 A Variant of the Cramer-Shoup Encryption Scheme

5.1 The (Twin) DDH Assumption

Let $\mathbb{G}$ be a group of order q and let g be a random generator. Distinguishing the two distributions $(X, Y, \mathrm{dh}(X,Y))$ and (X, Y, Z) for random $X, Y, Z \in \mathbb{G}$ is the [illegible] (DDH) problem. The [illegible] states that the DDH problem is hard. As a natural decision variant of the Twin DH problem, the [illegible] is distinguishing the two distributions $(X_1, X_2, Y, \mathrm{dh}(X_1, Y))$ and (X_1, X_2, Y, Z) for random $X_1, X_2, Y, Z \in \mathbb{G}$. The [illegible] states that the Twin DDH problem is hard, even given access to a decision oracle for the predicate for $\mathrm{2dhp}(X_1, X_2, \cdot, \cdot, \cdot)$, which on input $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ returns $\mathrm{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$. (Note the value $\mathrm{dh}(X_2, Y)$ is never provided as input to the distinguisher since otherwise the Strong Twin DDH assumption could trivially broken using the 2dhp oracle.)

We also consider a potentially weaker "hashed variants" of the above two assumptions. For a hash function $\mathsf{H} : \mathbb{G} \to \{0,1\}^\kappa$, the [illegible] problem is to distinguish the two distributions $(X, Y, \mathsf{H}(\mathrm{dh}(X,Y)))$ and (X, Y, k), for random $X, Y \in \mathbb{G}$ and $k \in \{0,1\}^\kappa$. The [illegible] states that the Hashed DDH problem is hard. In the same way, we can consider the Strong Twin Hashed DDH assumption.

We stress that the (Strong Twin) Hashed DDH assumption simplifies to the (Strong Twin) DDH assumption when H is the identity. Furthermore, there are natural groups (such as non-prime-order groups) where the DDH problem is known to be easy yet the Hashed DDH problem is still assumed to be hard for a reasonable choice of the hash function [12]. If H is modeled as random oracle then the Hashed DDH and the DH assumption become equivalent.

Using the trapdoor test in Theorem 2, we can prove an analog of Theorem 3.

Theorem 6. [illegible] $\mathcal{A}$ [illegible] Q_{d} [illegible] τ [illegible] $\mathcal{B}$ [illegible] $\mathcal{B}$ [illegible] τ [illegible] $O(Q_{\mathrm{d}} \log q)$ [illegible]

$$\mathsf{Adv2DDH}_{\mathcal{A},\mathbb{G}} \leq \mathsf{AdvDDH}_{\mathcal{B},\mathbb{G}} + Q_{\mathrm{d}}/q.$$

5.2 A Variant of the Cramer-Shoup Scheme

We now can consider the following encryption scheme which we call $\mathsf{PKE}_{\widetilde{\text{cs}}}$. This scheme makes use of a symmetric cipher (E, D) and a hash function $\mathsf{T} : \mathbb{G} \to \mathbb{Z}_q$ which we assume to be target collision-resistant [9]. A public key for this scheme is a tuple of random group elements $(X_1, \tilde{X}_1, X_2, \tilde{X}_2)$, with corresponding secret key $(x_1, \tilde{x}_1, x_2, \tilde{x}_2)$, where $X_i = g^{x_i}$ and $\tilde{X}_i = g^{\tilde{x}_i}$ for $i = 1, 2$. To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes

$$Y := g^y,\ t := \mathsf{T}(Y),\ Z_1 := (X_1^t \tilde{X}_1)^y,\ Z_2 := (X_2^t \tilde{X}_2)^y,\ k := \mathsf{H}(X_1^y),\ c := \mathsf{E}_k(m),$$

and the ciphertext is (Y, Z_1, Z_2, c). Decryption works as follows: given the ciphertext (Y, Z_1, Z_2, c), and secret key $(x_1, \tilde{x}_1, x_2, \tilde{x}_2)$, one computes $t := \mathsf{T}(Y)$ and checks if

$$Y^{x_1 t + \tilde{x}_1} = Z_1 \text{ and } Y^{x_2 t + \tilde{x}_2} = Z_2. \tag{5}$$

If not (we say the ciphertext is [illegible]), reject; otherwise, compute

$$k := \mathsf{H}(Y^{x_1}), \quad m := \mathsf{D}_k(c).$$

We remark that since $|\mathbb{G}| = |\mathbb{Z}_q| = q$, hash function T could be a bijection. See [6] for efficient constructions for certain groups $\mathbb{G}$.

Relation to Cramer-Shoup. Our scheme is very similar to the one by Cramer and Shoup [8]. Syntactically, the difference is that in Cramer-Shoup the value Z_1 is computed as $Z_1 = X_3^y$ (where X_3 is another random group element in the public key) and t is computed as $t = \mathsf{T}(Y, Z_1)$. However, our variant allows for a [illegible] based on the [illegible] DDH assumption whereas for the Cramer-Shoup scheme only a proof based on the DDH assumption is known (and the currently known proofs do not allow for it).

5.3 Security

We now show that, using the trapdoor test, $\mathsf{PKE}_{\widetilde{\text{cs}}}$ allows for a very elementary proof under the Hashed DDH assumption. We stress that are security proof is not in the random oracle model.

Theorem 7. [illegible] T [illegible] $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ [illegible] $\mathsf{PKE}_{\widetilde{\text{cs}}}$ [illegible]

[illegible] $\mathcal{A}$ [illegible] $\mathsf{PKE}_{\widetilde{\text{cs}}}$ [illegible] $\mathcal{A}$ [illegible] τ [illegible] Q_d [illegible] $\mathcal{B}_\text{ddh}$ [illegible] $\mathcal{B}_\text{sym}$ [illegible] SE [illegible] $\mathcal{B}_\text{tcr}$ [illegible] $\mathcal{B}_\text{ddh}$ $\mathcal{B}_\text{sym}$ [illegible] $\mathcal{B}_\text{tcr}$ [illegible] τ [illegible] $O(Q_\text{d} \log q)$ [illegible]

$$\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}_{\widetilde{\text{cs}}}} \leq \mathsf{AdvDDH}_{\mathcal{B}_\text{ddh},\mathbb{G},\mathsf{H}} + \mathsf{AdvCCA}_{\mathcal{B}_\text{sym},\mathsf{SE}} + \mathsf{AdvTCR}_{\mathcal{B}_\text{tcr},\mathsf{T}} + Q_\text{d}/q.$$

We proceed with a sequence of games.

Game 0. Let Game 0 be the original chosen ciphertext attack game, and let S_0 be the event that $\hat{b} = b$ in this game.

$$\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}_{\widetilde{\mathrm{cs}}}} = |\Pr[S_0] - 1/2|. \tag{6}$$

Game 1. Let Game 1 be like Game 0, but with the following difference. Game 1 aborts if the adversary, at any time, makes a decryption query containing a $\hat{Y}$ such that $\hat{Y} \neq Y$ and $\mathsf{T}(\hat{Y}) = \mathsf{T}(Y)$ where Y comes from the challenge ciphertext. Using a standard argument from [9] it is easy to show that

$$|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{AdvTCR}_{\mathcal{B}_{\mathrm{tcr}},\mathsf{T}}. \tag{7}$$

Game 2. Let Game 2 be as Game 1 with the following differences. For computing the public-key the experiment picks $x_1, x_2, y, a_1, a_2 \in \mathbb{Z}_q$ at random and computes $X_1 = g^{x_1}$, $X_2 = g^{x_2}$, and $Y = g^y$. Next, it computes $t := \mathsf{T}(Y)$ and

$$\tilde{X}_1 := X_1^{-t} g^{a_1}, \quad \tilde{X}_2 := X_2^{-t} g^{a_2}.$$

Note that the way the public-key is setup uses a technique to prove selective-ID security for IBE schemes [3].

The challenge ciphertext (Y, Z_1, Z_2, c) for message m_b is computed as

$$t := \mathsf{T}(Y), \quad Z_1 := Y^{a_1}, \quad Z_2 := Y^{a_2}, \quad k := \mathsf{H}(X_1^y), \quad c := \mathsf{E}_k(m_b). \tag{8}$$

This is a correctly distributed ciphertext for m_b and randomness $y = \log_g(Y)$ since, for $i = 1, 2$, $(X_i^t \tilde{X}_i)^y = (X_i^{t-t} g^{a_i})^y = (g^{a_i})^y = Y^{a_i} = Z_i$. We can assume (Y, Z_1, Z_2, k) to be computed in the beginning of the experiment since they are independent of m_0, m_1.

A decryption query for ciphertext $(\hat{Y}, \hat{Z}_1, \hat{Z}_2, \hat{c})$ is answered as follows. Compute $\hat{t} = \mathsf{T}(\hat{Y})$. If $t = \hat{t}$ then verify consistency by checking if $Z_1 = \hat{Z}_1$ and $Z_2 = \hat{Z}_2$. If the ciphertext is consistent then use the challenge key k defined in (8) to decrypt $\hat{c}$. If $t \neq \hat{t}$ then proceed as follows. For $i = 1, 2$, compute $\bar{Z}_i = (\hat{Z}_i / \hat{Y}^{a_i})^{1/(\hat{t}-t)}$. Consistency of the ciphertext is verified by checking if

$$\hat{Y}^{x_1} = \bar{Z}_1 \text{ and } \hat{Y}^{x_2} = \bar{Z}_2. \tag{9}$$

Let $\hat{y} = \log_g \hat{Y}$. The value $\hat{Z}_i$ was correctly generated iff $\hat{Z}_i = (X_i^{\hat{t}} \tilde{X}_i)^{\hat{y}} = (X_i^{\hat{t}-t} g^{a_i})^{\hat{y}} = (\hat{Y}^{x_i})^{\hat{t}-t} \cdot \hat{Y}^{a_i}$ which is equivalent to $\bar{Z}_i = \hat{Y}^{x_i}$. Hence, (9) is equivalent to the test from the original scheme (5). If the ciphertext is consistent then one can use the symmetric key $\hat{k} = \mathsf{H}(\bar{Z}_1) = \mathsf{H}(\hat{Y}^{x_1})$ to decrypt $\hat{c}$ and return $\hat{m} = \mathsf{D}_{\hat{k}}(\hat{c})$.

Let S_2 be the event that $\hat{b} = b$ in this game. As we have seen,

$$\Pr[S_2] = \Pr[S_1]. \tag{10}$$

Game 3. Let Game 3 be as Game 2 with the only difference that the value k to compute that challenge ciphertext is now chosen at random from $\mathbb{G}$. We claim that

$$|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv2DDH}_{\mathcal{B}_{2\mathrm{ddh}},\mathbb{G},\mathsf{H}}, \tag{11}$$

where $\mathcal{B}_{2\mathrm{ddh}}$ is an efficient Strong Twin Hashed DDH adversary that makes at most Q_d queries to the decision oracle. $\mathcal{B}_{2\mathrm{ddh}}$ is defined as follows. Using the values (X_1, X_2, Y, k) from its challenge (where either $k = \mathsf{H}(\mathrm{dh}(X_1, Y))$ or k is random), adversary $\mathcal{B}_{2\mathrm{ddh}}$ runs (without knowing x_1, x_2, y) the experiment as described in Game 2 using k as the challenge key in (8) to encrypt m_b. Note that the only point where Games 2 and 3 make use of x_1 and x_2 is the consistency check (9) which $\mathcal{B}_{2\mathrm{ddh}}$ equivalently implements using the 2dhp oracle, i.e. by checking if

$$2\mathrm{dhp}(X_1, X_2, \hat{Y}, \bar{Z}_1, \bar{Z}_2)$$

holds. We have that if $k = \mathsf{H}(\mathrm{dh}(X_1, Y)) \in \{0,1\}^\kappa$, this perfectly simulates Game 2, whereas if $k \in \{0,1\}^\kappa$ is random this perfectly simulates Game 3. This proves (11).

Finally, it is easy to see that in Game 3, the adversary is essentially playing the chosen ciphertext attack game against SE. Thus, there is an efficient adversary $\mathcal{B}_{\mathrm{sym}}$ such that

$$|\Pr[S_3] - 1/2| = \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}}. \tag{12}$$

The theorem now follows by combining (6)–(12) with Theorem 6.

5.4 A Variant with Security from the DH Assumption

We now consider a slight variant of the scheme $\mathsf{PKE}_{\widetilde{\mathrm{cs}}}$ that uses the Goldreich-Levin bit [14,13] to achieve security based on the (standard) DH assumption.

Let $\nu = O(\log \kappa)$ be some integer that divides the security parameter κ and set $\ell = \kappa/\nu$. Let the public key now contain the $2\ell + 3$ group elements Y and $X_i = g^{x_i}, \tilde{X}_i = g^{\tilde{x}_i}$, for $i = 1, \ldots, \ell + 1$. Furthermore, it contains a sufficiently large random bit-strings R to extract the Diffie-Hellman hard-core bits (a string of length $\ell \cdot 2\kappa$ is sufficient). To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes $Y := g^y$ and $Z_i := (X_i^t \tilde{X}_i)^y$, for $i = 1, \ldots, \ell + 1$, where $t = \mathsf{T}(Y)$. As before, the function of $Z_{\ell+1}$ is the consistency check. From each of the ℓ unique Diffie-Hellman keys $k_i = \mathsf{H}(X_i^y) \in \{0,1\}^\kappa$ $(i = 1, \ldots, \ell)$ and parts of R we can now extract a $\nu = \kappa/\ell$ simultaneous hard-core bits $k_i' \in \{0,1\}^\nu$. Finally, a concatenation of all k_i' yields a k-bit symmetric key $k \in \{0,1\}^\kappa$ that is used to encrypt m as $c = \mathsf{E}_k(m)$. The ciphertext is $(Y, Z_1, \ldots, Z_{\ell+1}, c)$. Decryption first verifies consistency of $(Y, Z_1, \ldots, Z_{\ell+1})$ by checking if $Y^{x_i t + \tilde{x}_i} = Z_i$, for all $i = 1, \ldots, \ell + 1$. Then the key k is reconstructed from the unique Diffie-Hellman keys $k_i = \mathsf{H}(Y^{x_i})$ as in encryption.

For concreteness we can consider a security parameter of $\kappa = 128$ bits and set $\nu = \log_2(\kappa) = 7$, which means the ciphertext overhead consists of $128/7 + 2 \approx 20$

group elements which account for $20 \cdot 256 \approx 5000$ bits when implemented on elliptic curves. Note that this is less than two standard RSA moduli for the same security level (3072 bits each, for $\kappa = 128$).

In the full version we show that the above scheme is chosen-ciphertext secure under the DH assumption. The proof uses a hybrid argument in connection with the trapdoor test from Theorem 2. Furthermore, it uses the Goldreich-Levin construction to extract $\nu = O(\log(\kappa))$ hard-core bits out of each Diffie-Hellman key. The security reduction is polynomial-time but due to the generic hard-core construction it is not very tight.

6 Identity Based Encryption

In this section we show how to apply the trapdoor test in Theorem 2 to identity-based encryption in pairing groups. We give a bilinear version of the strong twin DH problem and show that it can be reduced to the standard bilinear DH problem. We then use this assumption to construct a new IBE scheme that we call twin Boneh-Franklin below. The end result is a chosen ciphertext secure IBE scheme based on bilinear DH with one group element of overhead in the ciphertexts and a tighter reduction than the original scheme on which it is based.

6.1 A New Bilinear Assumption

In groups equipped with a pairing $: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, we can define the function

$$\mathrm{bdh}(X, Y, W) := Z, \quad \text{where } X = g^x,\ Y = g^y,\ W = g^w, \text{ and } Z = (g, g)^{wxy}.$$

Computing $\mathrm{bdh}(X, Y, W)$ using random $X, Y, W \in \mathbb{G}$ is the ( ), . The states that computing the BDH problem is hard. We define a predicate

$$\mathrm{bdhp}(X, \hat{Y}, \hat{W}, \hat{Z}) := \mathrm{bdh}(X, \hat{Y}, \hat{W}) \stackrel{?}{=} \hat{Z}.$$

We can also consider the BDH problem where, in addition to random (X, Y, W), one is also given access to an oracle that on input $(\hat{Y}, \hat{W}, \hat{Z})$ returns $\mathrm{bdhp}(X, \hat{Y}, \hat{W}, \hat{Z})$. The [22] states that the BDH problem remains hard even with the help of the oracle.

For reasons similar to the issue with hashed ElGamal encryption, the strong BDH assumption seems necessary to prove the CCA security of the basic version [22] of the original Boneh-Franklin IBE [5]. We can repeat the above idea and define the , where one must compute $2\mathrm{bdh}(X_1, X_2, Y, W)$ for random X_1, X_2, Y, W, where we define

$$2\mathrm{bdh}(X_1, X_2, Y, W) := (\mathrm{bdh}(X_1, Y, W), \mathrm{bdh}(X_2, Y, W)).$$

Continuing as above, the is the same as the twin BDH problem but with a suitably defined decision oracle. In this case define the predicate

$$2\mathrm{bdhp}(X_1, X_2, \hat{Y}, \hat{W}, \hat{Z}_1, \hat{Z}_2) := 2\mathrm{bdh}(X_1, X_2, \hat{Y}, \hat{W}) \stackrel{?}{=} (\hat{Z}_1, \hat{Z}_2),$$

and the decision oracle takes input $(\hat{Y}, \hat{W}, \hat{Z}_1, \hat{Z}_2)$ and returns $\text{2bdhp}(X_1, X_2, \hat{Y}, \hat{W}, \hat{Z}_1, \hat{Z}_2)$. The [illegible] states that the BDH problem is still hard, even with access to the decision oracle.

Finally, we will need a slight generalization of the trapdoor test in Theorem 2. It is easy to check that the theorem is still true if the elements $\hat{Y}, \hat{Z}_1, \hat{Z}_2$ are in a [illegible] cyclic group of the same order (we will take them in the range group of the pairing). With this observation, we can prove an analog of Theorem 3.

Theorem 8. [illegible] $\mathcal{A}$ [illegible] Q_{d} [illegible] $\mathcal{T}$ [illegible] $\mathcal{B}$ [illegible] $\mathcal{B}$ [illegible] $\mathcal{T}$ [illegible] $O(Q_{\text{d}} \log q)$ [illegible]

$$\mathsf{Adv2BDH}_{\mathcal{A},\mathbb{G}} \leq \mathsf{AdvBDH}_{\mathcal{B},\mathbb{G}} + Q_{\text{d}}/q.$$

[illegible] $\mathcal{B}$ [illegible] $1 - 1/q$

6.2 Twin Boneh-Franklin

For model and security definitions of IBE we refer the reader to [5]. Theorem 8 admits a simple analysis of the following IBE scheme, which we call the [illegible]. This scheme will use two hash functions, H (which outputs symmetric keys) and G (which outputs group elements), and a symmetric cipher (E, D). A system public key is a pair of group elements (X_1, X_2), where $X_i = g^{x_i}$ for $i = 1, 2$. The system private key is (x_1, x_2), which are selected at random from $\mathbb{Z}_q$ by the setup algorithm. The secret key for an identity [illegible] $\in \{0,1\}^*$ is $(S_1, S_2) = (\mathsf{G}(\ldots)^{x_1}, \mathsf{G}(\ldots)^{x_2})$. To encrypt a message m for identity [illegible], one chooses $y \in \mathbb{Z}_q$ and random and sets

$$Y := g^y, \quad Z_1 := \ (\mathsf{G}(\ldots), X_1)^y, \quad Z_2 := \ (\mathsf{G}(\ldots), X_2)^y,$$
$$k := \mathsf{H}(\ldots, Y, Z_1, Z_2), \quad c := \mathsf{E}_k(m).$$

The ciphertext is (Y, c). To decrypt using the secret key (S_1, S_2) for [illegible], one computes

$$Z_1 := \ (S_1, Y), \quad Z_2 := \ (S_2, Y), \quad k := \mathsf{H}(\ldots, Y, Z_1, Z_2), \quad m := \mathsf{D}_k(c).$$

We shall denote this scheme $\mathsf{IBE}_{\text{2dh}}$. Now we can essentially borrow the analysis of the original Boneh-Franklin scheme under the strong BDH assumption [22], except now we prove that the scheme is secure against [illegible] under the strong twin BDH assumption. By Theorem 8, we get that the above IBE scheme is CCA secure under the BDH assumption if the symmetric cipher is secure and the hash functions are treated as random oracles. The security reduction here enjoys the same tightness as the reduction given in [22], which is tighter than the original analysis of the Boneh-Franklin scheme. Again, for space reasons we will give a complete statement of this result and the corresponding proof (which is mostly standard) in the full version.

We remark that our ideas can also be applied to the IBE scheme from Sakai-Kasahara [27]. The resulting IBE scheme is more efficient but its security can only be proved based on the q-BDHI assumption [4].

Acknowledgments

We thank Masayuki Abe and Tatsuaki Okamoto for interesting discussions.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
3. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
5. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS 2005, pp. 320–329. ACM Press, New York (2005)
7. Coron, J.-S., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: GEM: A generic chosen-ciphertext secure encryption method. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 263–276. Springer, Heidelberg (2002)
8. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
9. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
10. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
11. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
12. Gennaro, R., Krawczyk, H., Rabin, T.: Secure Hashed Diffie-Hellman over non-DDH groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 361–381. Springer, Heidelberg (2004)
13. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge (2001)
14. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC, pp. 25–32. ACM Press, New York (1989)

15. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
16. Hanaoka, G., Kurosawa, K.: Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. Unpublished manuscript (2008)
17. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
18. Kim, K., Baek, J., Lee, B.: Secure length-saving ElGamal encryption under the computational Diffie-Hellman assumption. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 49–58. Springer, Heidelberg (2000)
19. Kudla, C., Paterson, K.G.: Modular security proofs for key agreement protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565. Springer, Heidelberg (2005)
20. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
21. Kurosawa, K., Matsuo, T.: How to remove MAC from DHIES. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 236–247. Springer, Heidelberg (2004)
22. Libert, B., Quisquater, J.-J.: Identity based encryption without redundancy. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 285–300. Springer, Heidelberg (2005)
23. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. In: The CRC Press series on discrete mathematics and its applications, CRC Press, Boca Raton (1997)
24. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
25. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)
26. Phan, D.H., Pointcheval, D.: About the security of ciphers (semantic security and pseudo-random permutations). In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 182–197. Springer, Heidelberg (2004)
27. Sakai, R., Kasahara, M.: ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054 (2003), http://eprint.iacr.org/
28. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
29. Steinfeld, R., Baek, J., Zheng, Y.: On the necessity of strong assumptions for the security of a class of asymmetric encryption schemes. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 241–256. Springer, Heidelberg (2002)

Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products

Jonathan Katz[1,⋆], Amit Sahai[2,⋆⋆], and Brent Waters[3,⋆⋆⋆]

[1] University of Maryland
jkatz@cs.umd.edu
[2] UCLA
sahai@cs.ucla.edu
[3] SRI International
bwaters@csl.sri.com

Abstract. Predicate encryption is a new paradigm generalizing, among other things, identity-based encryption. In a predicate encryption scheme, secret keys correspond to predicates and ciphertexts are associated with attributes; the secret key SK_f corresponding to a predicate f can be used to decrypt a ciphertext associated with attribute I if and only if $f(I) = 1$. Constructions of such schemes are currently known for relatively few classes of predicates.

We construct such a scheme for predicates corresponding to the evaluation of *inner products* over $\mathbb{Z}_N$ (for some large integer N). This, in turn, enables constructions in which predicates correspond to the evaluation of disjunctions, polynomials, CNF/DNF formulae, or threshold predicates (among others). Besides serving as a significant step forward in the theory of predicate encryption, our results lead to a number of applications that are interesting in their own right.

1 Introduction

Traditional public-key encryption is rather coarse-grained: a sender encrypts a message M with respect to a given public key PK, and only the owner of the (unique) secret key associated with PK can decrypt the resulting ciphertext and recover the message. These straightforward semantics suffice for point-to-point communication, where encrypted data is intended for one particular user who is known to the sender in advance. In other settings, however, the sender may

⋆ Research supported in part by NSF CAREER award #0447075 and the U.S. Army Research Laboratory.

⋆⋆ Supported in part by the NSF ITR and CyberTrust programs (including grants 0627781, 0456717, 0716389, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Okawa Research Award, and an Alfred P. Sloan Foundation Research Fellowship.

⋆⋆⋆ Supported by NSF CNS-0524252, CNS-0716199; the US Army Research Office under the CyberTA Grant No. W911NF-06-1-0316; and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 146–162, 2008.

instead want to define some complex determining who is allowed to recover the encrypted data. For example, classified data might be associated with certain keywords; this data should be accessible to users who are allowed to read classified information, as well as to users allowed to read information associated with the particular keywords in question. Or, in a health care application, a patient's records should perhaps be accessible only to a physician who has treated the patient in the past.

Applications such as those sketched above require new cryptographic mechanisms that provide more fine-grained control over access to encrypted data. offers one such tool. At a high level (formal definitions are given in Section 2), secret keys in a predicate encryption scheme correspond to predicates in some class $\mathcal{F}$, and a sender associates a ciphertext with an attribute in a set Σ; a ciphertext associated with the attribute $I \in \Sigma$ can be decrypted by a secret key SK_f corresponding to the predicate $f \in \mathcal{F}$ if and only if $f(I) = 1$.

The "basic" level of security achieved by such schemes guarantees, informally, that a ciphertext associated with attribute I hides all information about the underlying message unless one holds a secret key giving the explicit ability to decrypt. I.e., if an adversary $\mathcal{A}$ holds keys $SK_{f_1}, \ldots, SK_{f_\ell}$, then $\mathcal{A}$ learns nothing about a message encrypted using attribute I if $f_1(I) = \cdots = f_\ell(I) = 0$. We refer to this security notion as . A stronger notion of security, that we call , requires further that a ciphertext hides all information about the associated attribute I except that which is explicitly leaked by the keys in one's possession; i.e., an adversary holding secret keys as above learns only the values $f_1(I), \ldots, f_\ell(I)$ (in addition to the message in case one of these evaluates to '1'). Formal definitions are given in Section 2.

Much recent work aimed at constructing different types of fine-grained encryption schemes can be cast in the framework of predicate encryption. Identity-based encryption (IBE) [21,8,13,4,5,23] can be viewed as predicate encryption for the class of equality tests; the standard notion of security for IBE [8,12] corresponds to payload-hiding, while IBE [7,11,14] corresponds to the stronger notion of attribute hiding. Attribute-based encryption schemes [20,15,3,19] can also be cast in the framework of predicate encryption, though in this case all the listed constructions achieve payload hiding only. Boneh and Waters [10] construct a predicate encryption scheme that handles conjunctions (of, e.g., equality tests) and range queries; their scheme satisfies the stronger notion of attribute hiding. Shi et al. [22] also construct a scheme for range queries, but prove security in a weaker model where attribute hiding is required to hold only if the adversary holds keys that do allow recovery of the message.

Other work introducing concepts related to predicate encryption includes [2,1]. In contrast to the present work, however, the threat model in those works do not consider among users holding different secret keys. The problem becomes significantly more difficult when security against collusion is required.

1.1 Our Results

An important research direction is to construct predicate encryption schemes for predicate classes $\mathcal{F}$ that are as expressive as possible, with the ultimate goal being to handle all polynomial-time predicates. We are still far from this goal. Furthermore, most of the existing work (listed above) yields only payload-hiding schemes, and existing techniques for obtaining attribute-hiding schemes seem limited to enforcing [illegible] (indeed, handling disjunctions was left as an open question in [10]). Getting slightly technical, this is because the underlying cryptographic mechanism used in the above schemes is to pair components of the secret key with corresponding components of the ciphertext and then multiply the intermediate results together; a "cancelation" occurs if everything "matches", but a random group element results if there is any "mismatch". Thus, the holder of a non-matching secret key learns only that there was a mismatch in [illegible] position, but does not learn the number of mismatches or their locations. Very different cryptographic techniques seem needed to support disjunctions, since a mismatch in a single position cannot result in a completely random group element but rather must somehow allow for a "cancelation" if there is a match in some other position. (We stress that what makes this difficult is that attribute hiding requires correct decryption to hide the position of a match and only reveal that there was a match in at least one position.)

The aim of our work is to construct an attribute-hiding scheme handling disjunctions. As a stepping stone toward this goal, we first focus on predicates corresponding to the computation of inner products over $\mathbb{Z}_N$ (for some large integer N). Formally, we take $\Sigma = \mathbb{Z}_N^n$ as our set of attributes, and take our class of predicates to be $\mathcal{F} = \{f_{\boldsymbol{x}} \mid \boldsymbol{x} \in \mathbb{Z}_N^n\}$ where $f_{\boldsymbol{x}}(\boldsymbol{y}) = 1$ iff $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. (Here, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes the dot product $\sum_{i=1}^{n} x_i \cdot y_i \bmod N$ of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.) We construct a predicate encryption scheme for this $\mathcal{F}$ without random oracles, based on two new assumptions in composite-order groups equipped with a bilinear map. Our assumptions are non-interactive and of fixed size (i.e., not "q-type"), and can be shown to hold in the generic group model. A pessimistic interpretation of our results would be that we prove security in the generic group model, but we believe it is of importance that we are able to distill our necessary assumptions to ones that are compact and falsifiable. Our construction uses new techniques, including the fact that we work in a bilinear group whose order is a product of [illegible] primes.

We view our main construction as a significant step toward increasing the expressiveness of predicate encryption. Moreover, we show that any predicate encryption scheme supporting "inner product" predicates as described above can be used as a building block to construct predicates of more general types:

- As an easy warm-up, we show that it implies (anonymous) identity-based encryption as well as hidden-vector encryption [10]. As a consequence, our work implies all the results of [10].
- We can also construct predicate encryption schemes supporting polynomial evaluation. Here, we take $\mathbb{Z}_N$ as our set of attributes, and predicates correspond to polynomials in $\mathbb{Z}_N[x]$ of some bounded degree; a predicate evaluates

to 1 iff the corresponding polynomial evaluates to 0 on the point in question. We can also extend this to include multi-variate polynomials (in some bounded number of variables). A "dual" of this construction allows the attributes to be polynomials, and the predicates to correspond to evaluation at a fixed point.

– Given the above, we can fairly easily support predicates that are of other predicates (e.g., equality), thus achieving our main goal. In the context of identity-based encryption, this gives the ability to issue secret keys corresponding to a of identities that enables decryption whenever a ciphertext is encrypted to any identity in this set (without leaking which identity was actually used to encrypt).
– We also show how to handle predicates corresponding to DNF and CNF formulae of some bounded size.
– Working directly with our "inner product" construction, we can derive a scheme supporting threshold queries of the following form: Attributes are subsets of $A = \{1, \ldots, \ell\}$, and predicates take the form $\{f_{S,t} \mid S \subseteq A\}$ where $f_{S,t}(S') = 1$ iff $S \cap S' = t$. This is useful in the "fuzzy IBE" setting of Sahai and Waters [20], and improves on their work in that we achieve attribute hiding (rather than only payload hiding) and handle thresholds.

We defer further discussion regarding the above until Section 5.

2 Definitions

We formally define the syntax of predicate encryption and the security properties discussed informally in the Introduction. (Our definitions follow the general framework of those given in [10].) Throughout this section, we consider the general case where Σ denotes an arbitrary set of attributes and $\mathcal{F}$ denotes an arbitrary set of predicates over Σ. Formally, both Σ and $\mathcal{F}$ may depend on the security parameter and/or the master public parameters; for simplicity, we leave this implicit.

Definition 1. predicate encryption scheme $\mathcal{F}$ Σ PPT Setup, GenKey, Enc, Dec

- Setup 1^n ( ) PK ( ) SK
- GenKey SK ( ) $f \in \mathcal{F}$ SK_f
- Enc PK $I \in \Sigma$ M C $C \leftarrow \mathsf{Enc}_{PK}(I, M)$
- Dec SK_f C M $\perp$

n (PK, SK) $\mathsf{Setup}(1^n)$ $f \in \mathcal{F}$ $SK_f \leftarrow \mathsf{GenKey}_{SK}(f)$ $I \in \Sigma$

- $f(I) = 1$ ... $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I, M)) = M$
- $f(I) = 0$ ... $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I, M)) = \perp$ ...

We will also consider a variant of the above that we call a ... . Here, Enc takes only an attribute I (and no message); the correctness requirement is that $\mathsf{Dec}_{SK_f}(\mathsf{Enc}_{PK}(I)) = f(I)$ and so all the receiver learns is whether the predicate is satisfied. A predicate-only scheme can serve as a useful building block toward a full-fledged predicate encryption scheme.

Our definition of attribute-hiding security corresponds to the notion described informally earlier. Here, an adversary may request keys corresponding to the predicates $f_1, \ldots, f_\ell$ and is then given either $\mathsf{Enc}_{PK}(I_0, M_0)$ or $\mathsf{Enc}_{PK}(I_1, M_1)$ for attributes I_0, I_1 such that $f_i(I_0) = f_i(I_1)$ for all i. Furthermore, if $M_0 \neq M_1$ then it is required that $f_i(I_0) = f_i(I_1) = 0$ for all i. The goal of the adversary is to determine which attribute-message pair was encrypted, and the stated conditions ensure that this is not trivial. Observe that when specialized to the case when $\mathcal{F}$ consists of equality tests on strings, this notion corresponds to ... identity-based encryption (with selective-ID security).

Definition 2. ... $\mathcal{F}$ ... Σ ... attribute hiding (... secure) ... PPT ... $\mathcal{A}$ ... $\mathcal{A}$ ... n

$\mathcal{A}(1^n)$ ... $I_0, I_1 \in \Sigma$

$\mathsf{Setup}(1^n)$ ... PK, SK ... PK

$\mathcal{A}$ ... $f_1, \ldots, f_\ell \in \mathcal{F}$ ... $f_i(I_0) = f_i(I_1)$ ... i ... $\mathcal{A}$ ... $SK_{f_i} \leftarrow \mathsf{GenKey}_{SK}(f_i)$

$\mathcal{A}$ ... M_0, M_1 ... i ... $f_i(I_0) = f_i(I_1) = 1$ ... $M_0 = M_1$

... b ... $\mathcal{A}$ ... $C \leftarrow \mathsf{Enc}_{PK}(I_b, M_b)$

$\mathcal{A}$ ... b' ... $b' = b$

... $\mathcal{A}$ ... 1/2

We remark that our definition uses the “selective” notion of security introduced in [12]. One could also consider the definition where the adversary is allowed to specify I_0, I_1 adaptively, based on PK and any secret keys it obtains.

..., a strictly weaker notion of security, is defined by forcing $I_0 = I_1 = I$ in the above (in which case $\mathcal{A}$ has no possible advantage if $f_i(I) = 1$ for some i). For predicate-only encryption schemes we simply omit the messages in the above experiment. For convenience, we include in Appendix A a re-statement of the definition of security given above for the particular inner-product predicate we use in our main construction.

3 Background on Pairings and Complexity Assumptions

We review some general notions about bilinear groups of [illegible], first used in cryptographic applications by [9]. In contrast to all prior work using composite-order bilinear groups, however, we use groups whose order N is a product of [illegible] (distinct) primes. This is for simplicity only, since a variant of our construction can be proven secure based on a "decisional linear"-type assumption [6] in a group of composite order N which is a product of two primes.[1]

Let $\mathcal{G}$ be an algorithm that takes as input a security parameter 1^n and outputs a tuple $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ where p, q, r are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $N = pqr$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is [illegible] (i.e., $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$ we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$) and [illegible] (i.e., if g generates $\mathbb{G}$ then $\hat{e}(g, g)$ generates $\mathbb{G}_T$). We assume the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map $\hat{e}$ are all computable in time polynomial in n. Furthermore, we assume that the description of $\mathbb{G}$ and $\mathbb{G}_T$ includes generators of these groups.

We use the notation $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r$ to denote the subgroups of $\mathbb{G}$ having order p, q, and r, respectively. Observe that $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Note also that if g is a generator of $\mathbb{G}$, then the element g^{pq} is a generator of $\mathbb{G}_r$; the element g^{pr} is a generator of $\mathbb{G}_q$; and the element g^{qr} is a generator of $\mathbb{G}_p$. Furthermore, if, e.g., $h_p \in \mathbb{G}_p$ and $h_q \in \mathbb{G}_q$ then

$$\hat{e}(h_p, h_q) = \hat{e}\left((g^{qr})^{\alpha_1}, (g^{pr})^{\alpha_2}\right) = \hat{e}\left(g^{\alpha_1}, g^{r\alpha_2}\right)^{pqr} = 1,$$

where $\alpha_1 = \log_{g^{qr}} h_p$ and $\alpha_2 = \log_{g^{pr}} h_q$. Similar rules hold whenever $\hat{e}$ is applied to non-identity elements in distinct subgroups.

3.1 Our Assumptions

We now state the assumptions we use to prove security of our construction. As remarked earlier, these assumptions are new but we justify them by proving that they hold in the generic group model under the assumption that finding a non-trivial factor of N (the group order) is hard. (Details appear in the full version of this paper [17].) At a minimum, then, our construction can be viewed as secure in the generic group model. Nevertheless, we state our assumptions explicitly and highlight that they are non-interactive and of fixed size.

Assumption 1. Let $\mathcal{G}$ be as above. We say that $\mathcal{G}$ satisfies Assumption 1 if the advantage of any PPT algorithm $\mathcal{A}$ in the following experiment is negligible in the security parameter n:

1. $\mathcal{G}(1^n)$ is run to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$. Set $N = pqr$, and let g_p, g_q, g_r be generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively.

[1] This is analogous to the "folklore" transformation (see, e.g., [16]) that converts a scheme based on a group whose order N is a product of two primes to a scheme that uses a prime-order group. Typically, using composite order groups gives a simpler scheme; since the group sizes are larger, however, group operations are less efficient.

2. Choose random $Q_1, Q_2, Q_3 \in \mathbb{G}_q$, random $R_1, R_2, R_3 \in \mathbb{G}_r$, random $a, b, s \in \mathbb{Z}_p$, and a random bit ν. Give to $\mathcal{A}$ the values $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ as well as

$$g_p, \quad g_r, \quad g_q R_1, \quad g_p^b, \quad g_p^{b^2}, \quad g_p^a g_q, \quad g_p^{ab} Q_1, \quad g_p^s, \quad g_p^{bs} Q_2 R_2.$$

If $\nu = 0$ give $\mathcal{A}$ the value $T = g_p^{b^2 s} R_3$, while if $\nu = 1$ give $\mathcal{A}$ the value $T = g_p^{b^2 s} Q_3 R_3$.
3. $\mathcal{A}$ outputs a bit ν', and succeeds if $\nu' = \nu$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and 1/2.

Assumption 2. Let $\mathcal{G}$ be as above. We say that $\mathcal{G}$ satisfies Assumption 2 if the advantage of any PPT algorithm $\mathcal{A}$ in the following experiment is negligible in the security parameter n:

1. $\mathcal{G}(1^n)$ is run to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$. Set $N = pqr$, and let g_p, g_q, g_r be generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively.
2. Choose random $h \in \mathbb{G}_p$ and $Q_1, Q_2 \in \mathbb{G}_q$, random $s, \gamma \in \mathbb{Z}_q$, and a random bit ν. Give to $\mathcal{A}$ the values $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ as well as

$$g_p, \quad g_q, \quad g_r, \quad h, \quad g_p^s, \quad h^s Q_1, \quad g_p^\gamma Q_2, \quad \hat{e}(g_p, h)^\gamma.$$

If $\nu = 0$ then give $\mathcal{A}$ the value $\hat{e}(g_p, h)^{\gamma s}$, while if $\nu = 1$ then give $\mathcal{A}$ a random element of $\mathbb{G}_T$.
3. $\mathcal{A}$ outputs a bit ν', and succeeds if $\nu' = \nu$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and 1/2.

Note that both the above assumptions imply the hardness of factoring N.

4 Our Main Construction

Our main construction is a predicate-only scheme where the set of attributes is $\Sigma = \mathbb{Z}_N^n$, and the class of predicates is $\mathcal{F} = \{f_{\boldsymbol{x}} \mid \boldsymbol{x} \in \mathbb{Z}_N^n\}$ with $f_{\boldsymbol{x}}(\boldsymbol{y}) = 1$ iff $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0 \bmod N$. (We use vectors of length n for convenience only.) In this section we construct a predicate-only scheme and give some intuition about our proof. In Appendix B we show how our scheme can be extended to give a full-fledged predicate encryption scheme. All proofs of security appear in the full version of our paper [17].

Intuition. In our construction, each ciphertext has associated with it a (secret) vector $\boldsymbol{x}$, and each secret key corresponds to a vector $\boldsymbol{v}$. The decryption procedure must check whether $\boldsymbol{x} \cdot \boldsymbol{v} = 0$, and reveal nothing about $\boldsymbol{x}$ but whether this is true. To do this, we will make use of a bilinear group $\mathbb{G}$ whose order N is the product of three primes p, q, and r. Let $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$ denote the subgroups of $\mathbb{G}$ having order p, q, and r, respectively. We will (informally) assume, as in [9], that a random element in any of these subgroups is indistinguishable from a random element of $\mathbb{G}$.[2] Thus, we can use random elements from one subgroup to "mask" elements from another subgroup.

At a high level, we will use these subgroups as follows. $\mathbb{G}_q$ will be used to encode the vectors $\boldsymbol{x}$ and $\boldsymbol{v}$ in the ciphertext and secret keys, respectively. Computation of the inner product $\langle \boldsymbol{v}, \boldsymbol{x} \rangle$ will be done in $\mathbb{G}_q$ (in the exponent), using the bilinear map. $\mathbb{G}_p$ will be used to encode an equation (again in the exponent) that evaluates to zero when decryption is done properly. This subgroup is used to prevent an adversary from improperly "manipulating" the computation (by, e.g., changing the ordering of components of the ciphertext or secret key, raising these components to some power, etc.). On an intuitive level, if the adversary tries to manipulate the computation in any way, then the computation occurring in the $\mathbb{G}_p$ subgroup will no longer yield the identity (i.e., will no longer yield 0 in the exponent), but will instead have the effect of "masking" the correct answer with a random element of $\mathbb{G}_p$ (which will invalidate the entire computation). Elements in $\mathbb{G}_r$ are used for "general masking" of terms in other subgroups; i.e., random elements of $\mathbb{G}_r$ will be multiplied with various components of the ciphertext (and secret key) in order to "hide" information that might be present in the $\mathbb{G}_p$ and $\mathbb{G}_q$ subgroups.

We now proceed to the formal description of our scheme.

4.1 The Construction

$\mathsf{Setup}(1^n)$ The setup algorithm first runs $\mathcal{G}(1^n)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Next, it computes g_p, g_q, and g_r as generators of $\mathbb{G}_p, \mathbb{G}_q$, and $\mathbb{G}_r$, respectively. It then chooses $R_{1,i}, R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}, h_{2,i} \in \mathbb{G}_p$ uniformly at random for $i = 1$ to n, and $R_0 \in \mathbb{G}_r$ uniformly at random. The public parameters include $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with:

$$PK = \big(g_p, \quad g_r, \quad Q = g_q \cdot R_0, \quad \{H_{1,i} = h_{1,i} \cdot R_{1,i}, \quad H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1}^n\big) .$$

The master secret key SK is $\big(p, q, r, g_q, \{h_{1,i},\ h_{2,i}\}_{i=1}^n\big)$.

$\mathsf{Enc}_{PK}(\boldsymbol{x})$ Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{Z}_N$. This algorithm chooses random $s, \alpha, \beta \in \mathbb{Z}_N$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $i = 1$ to n. (Note: a random element $R \in \mathbb{G}_r$ can be sampled by choosing random $\delta \in \mathbb{Z}_N$ and setting $R = g_r^\delta$.) It outputs the ciphertext

$$C = \Big(C_0 = g_p^s, \quad \big\{\, C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, \quad C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i} \big\}_{i=1}^n \Big) .$$

$\mathsf{GenKey}_{SK}(\boldsymbol{v})$ Let $\boldsymbol{v} = (v_1, \ldots, v_n)$, and recall $SK = \big(p, q, r, g_q, \{h_{1,i},\ h_{2,i}\}_{i=1}^n\big)$. This algorithm chooses random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $i = 1$ to n, random $R_5 \in \mathbb{G}_r$, random $f_1, f_2 \in \mathbb{Z}_q$, and random $Q_6 \in \mathbb{G}_q$. It then outputs

$$SK_{\boldsymbol{v}} = \left(\begin{array}{c} K = R_5 \cdot Q_6 \cdot \prod_{i=1}^n h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \\ \Big\{ K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot v_i}, \quad K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot v_i} \Big\}_{i=1}^n \end{array} \right) .$$

[2] This is only for intuition. Our actual computational assumption is given in Section 3.

$\mathsf{Dec}_{SK_{\boldsymbol{v}}}(C)$ Let $C = \big(C_0, \{\, C_{1,i},\, C_{2,i}\}_{i=1}^n\big)$ and $SK_{\boldsymbol{v}} = \big(K, \{K_{1,i},\, K_{2,i}\}_{i=1}^n\big)$ be as above. The decryption algorithm outputs 1 iff

$$\hat{e}(C_0, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}) \stackrel{?}{=} 1.$$

Correctness. To see that correctness holds, let C and $SK_{\boldsymbol{v}}$ be as above. Then

$$\begin{aligned}
&\hat{e}(C_0, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}) \\
&= \hat{e}\left(g_p^s, \quad R_5 Q_6 \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \\
&\quad \cdot \prod_{i=1}^{n} \hat{e}\left(H_{1,i}^s Q^{\alpha \cdot x_i} R_{3,i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(H_{2,i}^s Q^{\beta \cdot x_i} R_{4,i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&= \hat{e}\left(g_p^s, \quad \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}\right) \\
&\quad \cdot \prod_{i=1}^{n} \hat{e}\left(h_{1,i}^s \cdot g_q^{\alpha \cdot x_i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i}\right) \cdot \hat{e}\left(h_{2,i}^s \cdot g_q^{\beta \cdot x_i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i}\right) \\
&= \prod_{i=1}^{n} \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) x_i v_i} \quad = \quad \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2 \bmod q)\langle \boldsymbol{x}, \boldsymbol{v}\rangle},
\end{aligned}$$

where α, β are random in $\mathbb{Z}_N$ and f_1, f_2 are random in $\mathbb{Z}_q$. If $\langle \boldsymbol{x}, \boldsymbol{v}\rangle = 0 \bmod N$, then the above evaluates to 1. If $\langle \boldsymbol{x}, \boldsymbol{v}\rangle \neq 0 \bmod N$ there are two cases: if $\langle \boldsymbol{x}, \boldsymbol{v}\rangle \neq 0 \bmod q$ then with all but negligible probability (over choice of α, β, f_1, f_2) the above evaluates to an element other than the identity. The other possibility is that $\langle \boldsymbol{x}, \boldsymbol{v}\rangle = 0 \bmod q$, in which case the above would always evaluate to 1; however, this would reveal a non-trivial factor of N and so this occurs with only negligible probability (recall, our assumptions imply hardness of finding a non-trivial factor of N).

There may appear to be some redundancy in our construction; for instance, the $C_{1,i}$ and $C_{2,i}$ components play almost identical roles. In fact we can view the encryption system as two parallel sub-systems linked via the C_0 component (and the corresponding component of the secret key). This two sub-system approach was first used by Boyen and Waters [11]; it can be viewed as a complex generalization of the Naor-Yung [18] "two-key" paradigm to the predicate encryption setting. A natural question is whether this redundancy can be eliminated to achieve better performance. While such a construction appears to be secure, our current proof (that utilizes a [illegible] assumption) relies in an essential way on having two parallel subsystems.

4.2 Proof Intuition

The most challenging aspect to providing a proof of our scheme naturally arises from the disjunctive capabilities of our system. In previous conjunctive systems (such as the one by Boneh and Waters [10]) the authors proved security by moving through a sequence of hybrid games, in which an encryption of a vector $\boldsymbol{x}$ was changed component-by-component to the encryption of a vector $\boldsymbol{y}$. The adversary could only ask for queries that did not match either $\boldsymbol{x}$ or $\boldsymbol{y}$, or queries that did not "look at" the components in which $\boldsymbol{x}$ and $\boldsymbol{y}$ differed. Thus, it was relatively straightforward to perform hybrid experiments over the components of $\boldsymbol{x}$ and $\boldsymbol{y}$ that differed, since the private keys given to the adversary did not "look at" these components.

In our proof an adversary will again try to determine whether a given ciphertext was encrypted with respect to $\boldsymbol{x}$ or $\boldsymbol{y}$. However, in our case the adversary can legally request a secret key for a vector $\boldsymbol{v}$ such that $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0$ and $\langle \boldsymbol{y}, \boldsymbol{v} \rangle = 0$; i.e., it may obtain a key that should enable correct decryption in either case. This means that we cannot use the same proof strategy as in previous, conjunctive schemes. For example, if we change just one component at a time, then the "hybrid" vector used in an intermediate step will likely not be orthogonal to $\boldsymbol{v}$ (and the adversary will be able to detect this because its secret key will no longer decrypt the given ciphertext). Therefore, we need to use a sequence of hybrid games in which entire vectors are changed in one step, instead of using a sequence of hybrid games where the vector is changed component-by-component.

To do this we take advantage of the fact that, as noted earlier, our encryption scheme contains two parallel sub-systems. In our proof we will use hybrid games where a challenge ciphertext will be encrypted with respect to one vector in the first sub-system and a [illegible] vector in the second sub-system. (Note that such a ciphertext is ill-formed, since any valid ciphertext will always use the same vector in each sub-system.) Let $(\boldsymbol{a}, \boldsymbol{b})$ denote a ciphertext encrypted using vector $\boldsymbol{a}$ in the first sub-system and $\boldsymbol{b}$ in the second sub-system. To prove indistinguishability when encrypting to $\boldsymbol{x}$ (which corresponds to $(\boldsymbol{x}, \boldsymbol{x})$) and when encrypting to $\boldsymbol{y}$ (which corresponds to $(\boldsymbol{y}, \boldsymbol{y})$), we will prove indistinguishability of the following sequence of hybrid games:

$$(\boldsymbol{x}, \boldsymbol{x}) \approx (\boldsymbol{x}, \boldsymbol{0}) \approx (\boldsymbol{x}, \boldsymbol{y}) \approx (\boldsymbol{0}, \boldsymbol{y}) \approx (\boldsymbol{y}, \boldsymbol{y}).$$

Using this structure in our proof allows us to use a simulator that will essentially be able to work in one sub-system without "knowing" what is happening in the other one. The simulator embeds a "subgroup decision-like" assumption into the challenge ciphertext for each experiment. The structure of the challenge will determine whether a sub-system encrypts the given vector or the zero vector. Details of our proof and further discussion are given in the full version of our paper [17].

5 Applications of Our Main Construction

In this section we discuss some applications of predicate encryption schemes of the type constructed in this paper. Our treatment here is general and can be based on any predicate encryption scheme supporting "inner product" queries; we do not rely on any specific details of our construction.

Given a vector $\boldsymbol{x} \in \mathbb{Z}_N^\ell$, we denote by $f_{\boldsymbol{x}} : \mathbb{Z}_N^\ell \rightarrow \{0,1\}$ the function such that $f_{\boldsymbol{x}}(\boldsymbol{y}) = 1$ iff $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. We define $\mathcal{F}_\ell \stackrel{\text{def}}{=} \{f_{\boldsymbol{x}} \mid \boldsymbol{x} \in \mathbb{Z}_N^\ell\}$. An [illegible] ℓ is an attribute-hiding predicate encryption scheme for the class of predicates $\mathcal{F}_\ell$.

5.1 Anonymous Identity-Based Encryption

As a warm-up, we show how anonymous identity-based encryption (IBE) can be recovered from any inner product encryption scheme of dimension 2. To generate the master public and secret keys for the IBE scheme, simply run the setup algorithm of the underlying inner product encryption scheme. To generate secret keys for the identity $I \in \mathbb{Z}_N$, set $\boldsymbol{I} := (1, I)$ and output the secret key for the predicate $f_{\boldsymbol{I}}$. To encrypt a message M for the identity $J \in \mathbb{Z}_N$, set $\boldsymbol{J}' := (-J, 1)$ and encrypt the message using the encryption algorithm of the underlying inner product encryption scheme and the attribute $\boldsymbol{J}'$. Since $\langle \boldsymbol{I}, \boldsymbol{J}' \rangle = 0$ iff $I = J$, correctness and security follow.

5.2 Hidden-Vector Encryption

Given a set Σ, let $\Sigma_\star = \Sigma \cup \{\star\}$. Hidden-vector encryption (HVE) [10] corresponds to a predicate encryption scheme for the class of predicates $\Phi_\ell^{\mathsf{hve}} = \{\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)} \mid a_1, \ldots, a_\ell \in \Sigma_\star\}$, where

$$\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = \begin{cases} 1 \text{ if, for all } i, \text{ either } a_i = x_i \text{ or } a_i = \star \\ 0 \text{ otherwise} \end{cases} .$$

A generalization of the ideas from the previous section can be used to realize hidden-vector encryption with $\Sigma = \mathbb{Z}_N$ from any inner product encryption scheme $(\mathsf{Setup}, \mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec})$ of dimension 2ℓ:

- The setup algorithm is unchanged.
- To generate a secret key corresponding to the predicate $\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}$, first construct a vector $\boldsymbol{A} = (A_1, \ldots, A_{2\ell})$ as follows:

$$\begin{aligned} &\text{if } a_i \neq \star: \ A_{2i-1} := 1, \quad A_{2i} := a_i \\ &\text{if } a_i = \star: \ A_{2i-1} := 0, \quad A_{2i} := 0. \end{aligned}$$

  Then output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\boldsymbol{A}})$.
- To encrypt a message M for the attribute $x = (x_1, \ldots, x_\ell)$, choose random $r_1, \ldots, r_\ell \in \mathbb{Z}_N$ and construct a vector $\boldsymbol{X_r} = (X_1, \ldots, X_{2\ell})$ as follows:

$$X_{2i-1} := -r_i \cdot x_i, \quad X_{2i} := r_i$$

  (where all multiplication is done modulo N). Then output the ciphertext $C \leftarrow \mathsf{Enc}_{PK}(\boldsymbol{X_r}, M)$.

To see that correctness holds, let $(a_1, \ldots, a_\ell)$, $\boldsymbol{A}$, $(x_1, \ldots, x_\ell)$, $\boldsymbol{r}$, and $\boldsymbol{X_r}$ be as above. Then:

$$\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = 1 \;\;\Rightarrow\;\; \forall \boldsymbol{r} : \langle \boldsymbol{A}, \boldsymbol{X_r} \rangle = 0 \;\;\Rightarrow\;\; \forall \boldsymbol{r} : f_{\boldsymbol{A}}(\boldsymbol{X_r}) = 1.$$

Furthermore, assuming $\gcd(a_i - x_i, N) = 1$ for all i:

$$\begin{aligned}\phi^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = 0 &\Rightarrow \Pr_{\boldsymbol{r}}\left[\langle \boldsymbol{A}, \boldsymbol{X_r} \rangle = 0\right] = 1/N \\ &\Rightarrow \Pr_{\boldsymbol{r}}\left[f_{\boldsymbol{A}}(\boldsymbol{X_r}) = 1\right] = 1/N,\end{aligned}$$

which is negligible. Using this, one can prove security of the construction as well.

A straightforward modification of the above gives a scheme that is the "dual" of HVE, where the set of attributes is $(\Sigma_\star)^\ell$ and the class of predicates is $\bar{\Phi}^{\mathsf{hve}}_\ell = \{\bar{\phi}^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)} \mid a_1, \ldots, a_\ell \in \Sigma\}$ with

$$\bar{\phi}^{\mathsf{hve}}_{(a_1,\ldots,a_\ell)}(x_1, \ldots, x_\ell) = \begin{cases} 1 \text{ if, for all } i, \text{ either } a_i = x_i \text{ or } x_i = \star \\ 0 \text{ otherwise} \end{cases}.$$

5.3 Predicate Encryption Schemes Supporting Polynomial Evaluation

We can also construct predicate encryption schemes for predicates corresponding to polynomial evaluation. Let $\Phi^{\mathrm{poly}}_{\leq d} = \{f_p \mid p \in \mathbb{Z}_N[x], \deg(p) \leq d\}$, where

$$\phi_p(x) = \begin{cases} 1 \text{ if } p(x) = 0 \\ 0 \text{ otherwise} \end{cases}$$

for $x \in \mathbb{Z}_N$. Given an inner product encryption scheme ($\mathsf{Setup}, \mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec}$) of dimension $d + 1$, we can construct a predicate encryption scheme for $\Phi^{\mathrm{poly}}_{\leq d}$ as follows:

- The setup algorithm is unchanged.
- To generate a secret key corresponding to the polynomial $p(x) = a_d x^d + \cdots + a_0 x^0$, set $\boldsymbol{p} := (a_d, \ldots, a_0)$ and output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\boldsymbol{p}})$.
- To encrypt a message M for the attribute $w \in \mathbb{Z}_N$, set $\boldsymbol{w} := (w^d \bmod N, \ldots, w^0 \bmod N)$ and output the ciphertext $C \leftarrow \mathsf{Enc}_{PK}(\boldsymbol{w}, M)$.

Since $p(w) = 0$ iff $\langle \boldsymbol{p}, \boldsymbol{w} \rangle = 0$, correctness and security follow.

The above shows that we can construct predicate encryption schemes where predicates correspond to univariate polynomials whose degree d is polynomial in the security parameter. This can be generalized to the case of polynomials in t variables, and degree at most d in each variable, as long as d^t is polynomial in the security parameter.

We can also construct schemes that are the "dual" of the above, in which attributes correspond to polynomials and predicates involve the evaluation of the input polynomial at some fixed point.

5.4 Disjunctions, Conjunctions, and Evaluating CNF and DNF Formulas

Given the polynomial-based constructions of the previous section, we can fairly easily build predicate encryption schemes for disjunctions of equality tests. For example, the predicate OR_{I_1,I_2}, where $\mathsf{OR}_{I_1,I_2}(x) = 1$ iff either $x = I_1$ or $x = I_2$, can be encoded as the univariate polynomial

$$p(x) = (x - I_1) \cdot (x - I_2),$$

which evaluates to 0 iff the relevant predicate evaluates to 1. Similarly, the predicate $\overline{\mathsf{OR}}_{I_1,I_2}$, where $\overline{\mathsf{OR}}_{I_1,I_2}(x_1, x_2) = 1$ iff either $x_1 = I_1$ or $x_2 = I_2$, can be encoded as the bivariate polynomial

$$p'(x_1, x_2) = (x_1 - I_1) \cdot (x_2 - I_2).$$

Conjunctions can be handled in a similar fashion. Consider, for example, the predicate AND_{I_1,I_2} where $\mathsf{AND}_{I_1,I_2}(x_1, x_1) = 1$ if both $x_1 = I_1$ and $x_2 = I_2$. Here, we determine the relevant secret key by choosing a random $r \in \mathbb{Z}_N$ and letting the secret key correspond to the polynomial

$$p''(x_1, x_2) = r \cdot (x_1 - I_1) + (x_2 - I_2).$$

If $\mathsf{AND}_{I_1,I_2}(x_1, x_1) = 1$ then $p''(x_1, x_2) = 0$, whereas if $\mathsf{AND}_{I_1,I_2}(x_1, x_1) = 0$ then, with all but negligible probability over choice of r, it will hold[3] that $p''(x_1, x_2) \neq 0$.

The above ideas extend to more complex combinations of disjunctions and conjunctions, and for boolean variables this means we can handle arbitrary CNF or DNF formulas. (For non-boolean variables we do not know how to directly handle negation.) As pointed out in the previous section, the complexity of the resulting scheme depends polynomially on d^t, where t is the number of variables and d is the maximum degree (of the resulting polynomial) in each variable.

5.5 Exact Thresholds

We conclude with an application that relies directly on inner product encryption. Here, we consider the setting of "fuzzy IBE" [20], which can be mapped to the predicate encryption framework as follows: fix a set $A = \{1, \ldots, \ell\}$ and let the set of attributes be all subsets of A. Predicates take the form $\Phi = \{\phi_S \mid S \subseteq A\}$ where $\phi_S(S') = 1$ iff $|S \cap S'| \geq t$, i.e., S and S' overlap in [illegible] t positions. Sahai and Waters [20] show a construction of a payload-hiding predicate encryption scheme for this class of predicates.

We can construct a scheme where the attribute space is the same as before, but the class of predicates corresponds to overlap in [illegible] t positions.

[3] In general, the secret key may leak the value of r in which case the adversary will be able to find x_1, x_2 such that $\mathsf{AND}_{I_1,I_2}(x_1, x_1) \neq 1$ yet $p\ (x_1, x_2) = 0$. Since, however, we consider the "selective" notion of security (where the adversary must commit to x_1, x_2 at the outset of the experiment), this is not a problem in our setting.

(Our scheme will also be attribute hiding.) Namely, set $\Phi' = \{\phi'_S \mid S \subseteq A\}$ with $\phi'_S(S') = 1$ iff $|S \cap S'| = t$. Then, given any inner product encryption scheme of dimension $\ell + 1$:

- The setup algorithm is unchanged.
- To generate a secret key for the predicate ϕ'_S, first define a vector $\boldsymbol{v} \in \mathbb{Z}_N^{\ell+1}$ as follows:

$$\text{for } 1 \le i \le \ell: \quad v_i = 1 \text{ iff } i \in S$$
$$v_{\ell+1} = 1.$$

  Then output the key obtained by running $\mathsf{GenKey}_{SK}(f_{\boldsymbol{v}})$.
- To encrypt a message M for the attribute $S' \subseteq A$, define a vector $\boldsymbol{v}'$ as follows:

$$\text{for } 1 \le i \le \ell: \quad v_i = 1 \text{ iff } i \in S'$$
$$v_{\ell+1} = -t \bmod N.$$

  Then output the ciphertext $C \leftarrow \mathsf{Enc}_{PK}(\boldsymbol{v}', M)$.

Since $|S \cap S'| = t$ exactly when $\langle \boldsymbol{v}, \boldsymbol{v}' \rangle = 0$, correctness and security follow.

Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

1. Al-Riyami, S., Malone-Lee, J., Smart, N.: Escrow-free encryption supporting cryptographic workflow. Intl. J. Information Security 5(4), 217–229 (2006)
2. Bagga, W., Molva, R.: Policy-based cryptography and applications. In: Financial Cryptography (2005)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy (2007)
4. Boneh, D., Boyen, X.: Efficient selective-ID identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, Springer, Heidelberg (2004)
5. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, Springer, Heidelberg (2004)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public-key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, Springer, Heidelberg (2004)
8. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM J. Computing 32(3), 586–615 (2003)
9. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Theory of Cryptography Conference (2005)
10. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Theory of Cryptography Conference (2007)

11. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
12. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, Springer, Heidelberg (2003)
13. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
14. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, Springer, Heidelberg (2006)
15. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCCS (2006)
16. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) Eurocrypt 2008. LNCS, vol. 4965, Springer, Heidelberg (2008)
17. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. Cryptology ePrint Archive, Report 2007/404
18. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437 (1990)
19. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM CCCS (2007)
20. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, Springer, Heidelberg (2005)
21. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, Springer, Heidelberg (1985)
22. Shi, E., Bethencourt, J., Chan, H.T.-H., Song, D.X., Perrig, A.: Multi-dimensional range queries over encrypted data. In: IEEE Symposium on Security and Privacy (2007)
23. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, Springer, Heidelberg (2005)

A Security Definition for Inner-Product Encryption

Here, we re-state Definition 2 in the particular setting of our main construction, which is a predicate-only scheme where the set of attributes[4] is $\Sigma = \mathbb{Z}_N^n$ and the class of predicates is $\mathcal{F} = \{f_{\boldsymbol{x}} \mid \boldsymbol{x} \in \mathbb{Z}_N^n\}$ such that $f_{\boldsymbol{x}}(\boldsymbol{y}) = 1 \Leftrightarrow \langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

Definition 3. [illegible] $\Sigma, \mathcal{F}$ [illegible] attribute-hiding [illegible] PPT [illegible] $\mathcal{A}$ [illegible] $\mathcal{A}$ [illegible] n

Setup(1^n) [illegible] PK, SK [illegible] N [illegible] $\mathcal{A}$

$\mathcal{A}$ [illegible] $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_N^n$ [illegible] PK

$\mathcal{A}$ [illegible] $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell \in \mathbb{Z}_N^n$ [illegible] i $\langle \boldsymbol{v}_i, \boldsymbol{x} \rangle = \langle \boldsymbol{v}_i, \boldsymbol{y} \rangle$ [illegible] $\mathcal{A}$ [illegible] $SK_{\boldsymbol{v}_i} \leftarrow \mathsf{GenKey}_{SK}(f_{\boldsymbol{v}_i})$

[4] Technically speaking, both Σ and $\mathcal{F}$ depend on the public parameters (since N is generated as part of PK), but we ignore this technicality.

b … $b = 0$ … $\mathcal{A}$ … $C \leftarrow \mathsf{Enc}_{PK}(\boldsymbol{x})$ … $b = 1$ … $\mathcal{A}$ … $C \leftarrow \mathsf{Enc}_{PK}(\boldsymbol{y})$

$\mathcal{A}$ … b' … $b' = b$

$\mathcal{A}$ … $1/2$

B A Full-Fledged Predicate Encryption Scheme

In Section 4, we showed a construction of a … scheme. Here, we extend the previous scheme to obtain a full-fledged predicate encryption scheme in the sense of Definition 1. The construction follows.

$\mathsf{Setup}(1^n)$ The setup algorithm first runs $\mathcal{G}(1^n)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Next, it computes g_p, g_q, and g_r as generators of $\mathbb{G}_p, \mathbb{G}_q$, and $\mathbb{G}_r$, respectively. It then chooses $R_{1,i}, R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}, h_{2,i} \in \mathbb{G}_p$ uniformly at random for $i = 1$ to n, and $R_0 \in \mathbb{G}_r$ uniformly at random. It also chooses random $\gamma \in \mathbb{Z}_p$ and $h \in \mathbb{G}_p$. The public parameters include $(N = pqr, \mathbb{G}, \mathbb{G}_T, \hat{e})$ along with:

$$PK = \left(\begin{array}{c} g_p, \quad g_r, \quad Q = g_q \cdot R_0, \quad P = \hat{e}(g_p, h)^{\gamma}, \\ \left\{ H_{1,i} = h_{1,i} \cdot R_{1,i}, \quad H_{2,i} = h_{2,i} \cdot R_{2,i} \right\}_{i=1}^{n} \end{array} \right).$$

(This is identical to the public parameters of the predicate-only scheme with the exception of the additional element $P = \hat{e}(g_p, h)^{\gamma}$.) The master secret key SK is $\left(p, q, r, g_q, h^{-\gamma}, \{h_{1,i},\ h_{2,i}\}_{i=1}^{n}\right)$.

$\mathsf{Enc}_{PK}(\boldsymbol{x}, M)$ View M as an element of $\mathbb{G}_T$, and let $\boldsymbol{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{Z}_N$. This algorithm chooses random $s, \alpha, \beta \in \mathbb{Z}_N$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $i = 1$ to n. It outputs the ciphertext

$$C = \left(\begin{array}{c} C' = M \cdot P^s, \quad C_1 = g_p^s, \\ \left\{ C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, \quad C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i} \right\}_{i=1}^{n} \end{array} \right).$$

$\mathsf{GenKey}_{SK}(\boldsymbol{v})$ Let $\boldsymbol{v} = (v_1, \ldots, v_n)$. This algorithm chooses random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $i = 1$ to n, random $f_1, f_2 \in \mathbb{Z}_q$, random $R_5 \in \mathbb{G}_r$, and random $Q_6 \in \mathbb{G}_q$. It then outputs

$$SK_{\boldsymbol{v}} = \left(\begin{array}{c} K = R_5 \cdot Q_6 \cdot h^{-\gamma} \cdot \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \\ \left\{ K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot v_i}, \quad K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot v_i} \right\}_{i=1}^{n} \end{array} \right).$$

$\mathsf{Dec}_{SK_{\boldsymbol{v}}}(C)$ Let C and $SK_{\boldsymbol{v}}$ be as above. The decryption algorithm outputs

$$C' \cdot \hat{e}(C_1, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}).$$

As we have described it, decryption never returns an error (i.e., even when $\langle \boldsymbol{v}, \boldsymbol{x} \rangle \neq 0$). We will show below that when $\langle \boldsymbol{v}, \boldsymbol{x} \rangle \neq 0$ then the output is essentially a random element in the order-q subgroup of $\mathbb{G}_T$. By restricting the message space to some efficiently-recognizable set of negligible density in this subgroup, we recover the desired semantics by returning an error if the recovered message does not lie in this space.

Correctness. Let C and $SK_{\boldsymbol{v}}$ be as above. Then

$$\begin{aligned}
&C' \cdot \hat{e}(C_1, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \cdot \hat{e}(C_{2,i}, K_{2,i}) \\
&= M \cdot P^s \cdot \hat{e}\left(g_p^s, \quad R_5 Q_6 h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}} \right) \\
&\quad \cdot \prod_{i=1}^{n} \hat{e}\left(H_{1,i}^s Q^{\alpha \cdot x_i} R_{3,i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i} \right) \cdot \hat{e}\left(H_{2,i}^s Q^{\beta \cdot x_i} R_{4,i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i} \right) \\
&= M \cdot P^s \cdot \hat{e}\left(g_p^s, \quad h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}} \right) \\
&\quad \cdot \prod_{i=1}^{n} \hat{e}\left(h_{1,i}^s \, g_q^{\alpha \cdot x_i}, \quad g_p^{r_{1,i}} g_q^{f_1 \cdot v_i} \right) \cdot \hat{e}\left(h_{2,i}^s \, g_q^{\beta \cdot x_i}, \quad g_p^{r_{2,i}} g_q^{f_2 \cdot v_i} \right) \\
&= M \cdot P^s \cdot \hat{e}(g_p, h)^{-\gamma s} \cdot \prod_{i=1}^{n} \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) x_i v_i} \quad = \quad M \cdot \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) \langle \boldsymbol{x}, \boldsymbol{v} \rangle}.
\end{aligned}$$

If $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0 \bmod N$, then the above evaluates to M. If $\langle \boldsymbol{x}, \boldsymbol{v} \rangle \neq 0 \bmod N$ there are two cases: if $\langle \boldsymbol{x}, \boldsymbol{v} \rangle \neq 0 \bmod q$ then the above evaluates to an element whose distribution is statistically close to uniform in the order-q subgroup of $\mathbb{G}_T$. (Recall that α, β are chosen at random.) It is possible that $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0 \bmod q$, in which case the above always evaluates to M; however, this reveals a non-trivial factor of N and so an adversary can cause this condition to occur with only negligible probability.

Isogenies and the Discrete Logarithm Problem in Jacobians of Genus 3 Hyperelliptic Curves

Benjamin Smith

INRIA Saclay–Île-de-France
Laboratoire d'Informatique de l'École polytechnique (LIX)
91128 Palaiseau Cedex, France
smith@lix.polytechnique.fr

Abstract. We describe the use of explicit isogenies to translate instances of the Discrete Logarithm Problem from Jacobians of hyperelliptic genus 3 curves to Jacobians of non-hyperelliptic genus 3 curves, where they are vulnerable to faster index calculus attacks. We provide explicit formulae for isogenies with kernel isomorphic to $(\mathbb{Z}/2\mathbb{Z})^3$ (over an algebraic closure of the base field) for any hyperelliptic genus 3 curve over a field of characteristic not 2 or 3. These isogenies are rational for a positive fraction of all hyperelliptic genus 3 curves defined over a finite field of characteristic $p > 3$. Subject to reasonable assumptions, our constructions give an explicit and efficient reduction of instances of the DLP from hyperelliptic to non-hyperelliptic Jacobians for around 18.57% of all hyperelliptic genus 3 curves over a given finite field.

1 Introduction

After the great success of elliptic curves in cryptography, researchers have naturally been drawn to their higher-dimensional generalizations: Jacobians of higher-genus curves. Curves of genus 1 (elliptic curves), 2, and 3 are widely believed to offer the best balance of security and efficiency. This article is concerned with the security of curves of genus 3.

There are two classes of curves of genus 3: hyperelliptic and non-hyperelliptic. Each class has a distinct geometry: the canonical morphism of a hyperelliptic curve is a double cover of a curve of genus 0, while the canonical morphism of a non-hyperelliptic curve of genus 3 is an isomorphism to a nonsingular plane quartic curve. A hyperelliptic curve cannot be isomorphic (or birational) to a non-hyperelliptic curve. From a cryptological point of view, the Discrete Logarithm Problem (DLP) in Jacobians of hyperelliptic curves of genus 3 over $\mathbb{F}_q$ may be solved in $\widetilde{O}(q^{4/3})$ group operations, using the index calculus algorithm of Gaudry, Thomé, Thériault, and Diem [6]. Jacobians of non-hyperelliptic curves of genus 3 over $\mathbb{F}_q$ are amenable to Diem's index calculus algorithm [3], which requires only $\widetilde{O}(q)$ group operations to solve the DLP (for comparison, Pollard/baby-step-giant-step methods require $\widetilde{O}(q^{3/2})$ group operations to solve the DLP in Jacobians of genus 3 curves over $\mathbb{F}_q$). The security of non-hyperelliptic genus 3 curves is therefore widely held to be lower than that of their hyperelliptic cousins.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 163–180, 2008.

Our aim is to provide a means of efficiently translating DLPs from Jacobians of hyperelliptic genus 3 curves to Jacobians of non-hyperelliptic curves, where faster index calculus is available. We do this by constructing explicit [illegible] of Jacobians: surjective homomorphisms, with finite kernel, from hyperelliptic to non-hyperelliptic Jacobians. The kernels of our isogenies will intersect trivially with any subgroup of cryptographic interest, and so the isogenies will restrict to isomorphisms of DLP subgroups.

Specifically, let H be a hyperelliptic curve of genus 3 over a finite field of characteristic $p > 3$. Suppose the Jacobian J_H of H contains a subgroup S isomorphic to $(\mathbb{Z}/2\mathbb{Z})^3$ (over an algebraic closure of the base field), generated by differences of Weierstrass points. If the 2-Weil pairing restricts trivially to S, then there exists an isogeny with kernel S from J_H to a principally polarized abelian variety A. Using Recillas' trigonal construction [12], A may be realized as the Jacobian of a genus 3 curve X. This construction appears to be due to Donagi and Livné [5]; our contribution, aside from the cryptological application, is to provide explicit formulae for the curve X and the isogeny. Naïve moduli space dimension arguments suggest that there is an overwhelming probability that X will be non-hyperelliptic, and thus explicitly isomorphic to a nonsingular plane quartic curve C. We therefore obtain an explicit isogeny $\phi : J_H \to J_C$ with kernel S. If ϕ is defined over $\mathbb{F}_q$, then it maps $J_H(\mathbb{F}_q)$ into $J_C(\mathbb{F}_q)$, where Diem's $\widetilde{O}(q)$ index calculus is available. Given points P and $Q = [n]P$ of odd order in $J_H(\mathbb{F}_q)$, we can solve the DLP (that is, recovering n from P and Q) in $J_C(\mathbb{F}_q)$, using

$$Q = [n]P \implies \phi(Q) = [n]\phi(P).$$

There are several caveats to our approach, besides the requirement of a subgroup S as described above. First, it does not apply in characteristic 2 or 3. In characteristic 2, the subgroup S is the kernel of a verschiebung, so X is necessarily hyperelliptic. In characteristic 3, we cannot use the trigonal construction. Second, in order to obtain an advantage with index calculus on X over H, the isogeny must be defined over $\mathbb{F}_q$ and X must be non-hyperelliptic. We show in §8 that, subject to some reasonable assumptions, given a hyperelliptic curve H of genus 3 over a sufficiently large finite field, our algorithms succeed in giving an explicit rational isogeny from J_H to a non-hyperelliptic Jacobian with probability ≈ 0.1857. In particular, instances of the DLP can be solved in $\widetilde{O}(q)$ group operations for around 18.57% of all Jacobians of hyperelliptic curves of genus 3 over a finite field of characteristic $p > 3$.

Our results have a number of interesting implications for curve-based cryptography, at least for curves of genus 3. First, the difficulty of the DLP in a subgroup G of J_H depends not only on the size of the subgroup G, but upon the existence of other rational subgroups of J_H that can be used to form quotients. Second, the security of a given hyperelliptic genus 3 curve depends significantly upon the factorization of its hyperelliptic polynomial. Neither of these results has any parallel in genus 1 or 2.

After reviewing some standard definitions for hyperelliptic curves in §2, we define the kernels of our isogenies in §3. In §4, §5 and §6, we describe and derive explicit formulae for the trigonal construction, which is our main tool for constructing isogenies. After giving an example in §7, we compute (heuristically) the expectation that the methods of this article will compute a rational isogeny for a randomly chosen curve in §8. Finally, in §9 we briefly describe some of the problems involved in generalizing these methods.

A Note on the Base Field

We will work over $\mathbb{F}_q$ throughout this article, where q is a power of a prime $p > 3$. We let $\mathcal{G}$ denote the Galois group $\mathrm{Gal}(\overline{\mathbb{F}}_q/\mathbb{F}_q)$, which is (topologically) generated by the q^{th} power Frobenius map. Some of the theory of this article carries over to fields of characteristic zero: in particular, the results of §5 and §6 are valid over fields of characteristic not 2 or 3.

2 Notation and Conventions for Hyperelliptic Curves

We assume that we are given a hyperelliptic curve H of genus 3 over $\mathbb{F}_q$, and that the Jacobian J_H of H is absolutely simple. We will use both an affine model

$$H : y^2 = F(x),$$

where F is a squarefree polynomial of degree 7 or 8, and a weighted projective plane model

$$H : w^2 = \widetilde{F}(u, v)$$

for H (where u, v, and w have weights 1, 1, and 4, respectively). The coordinates of these models are related by $x = u/v$ and $y = w/v^4$. The polynomial $\widetilde{F}$ is squarefree of total degree 8, with $\widetilde{F}(u, v) = v^8 F(u/v)$ and $\widetilde{F}(x, 1) = F(x)$. We emphasize that F need not be monic. By a [illegible], we mean the hyperelliptic curve defined by $w^2 = \widetilde{F}(u, v)$, where $\widetilde{F}$ is a uniformly randomly chosen squarefree homogenous bivariate polynomial of degree 8 over $\mathbb{F}_q$. The canonical [illegible] ι of H is defined by $(x, y) \mapsto (x, -y)$ in the affine model, $(u : v : w) \mapsto (u : v : -w)$ in the projective model, and induces the negation map $[-1]$ on J_H. The quotient $\pi : H \to \mathbb{P}^1 \cong H/\langle \iota \rangle$ sends $(u : v : w)$ to $(u : v)$ in the projective model, and (x, y) to x in the affine model (where it maps onto the affine patch of $\mathbb{P}^1$ where $v \neq 0$).

To compute in J_H, we fix an isomorphism from J_H to the group of degree-zero divisor classes on H, denoted $\mathrm{Pic}^0(H)$. Recall that divisors are formal sums of points in $H(\overline{\mathbb{F}_q})$, and if $D = \sum_{P \in H} n_P(P)$ is a divisor, then $\sum_{P \in H} n_P$ is the [illegible] of D. We say D is [illegible] if $D = \mathrm{div}(f) := \sum_{P \in H} \mathrm{ord}_P(f)(P)$ for some function f on H, where $\mathrm{ord}_P(f)$ denotes the number of zeroes (or the negative of the number of poles) of f at P. Since H is complete, every principal divisor has degree 0. The group $\mathrm{Pic}^0(H)$ is defined to be the group of divisors of degree 0 modulo principal divisors; the equivalence class of a divisor D is denoted by $[D]$.

3 The Kernel of the Isogeny

The eight points of $H(\overline{\mathbb{F}_q})$ where $w = 0$ are called the [illegible] of H. Each Weierstrass point W corresponds to a linear factor $L_W = v(W)u - u(W)v$ of $\widetilde{F}$. If W_1 and W_2 are Weierstrass points, then $2(W_1) - 2(W_2) = \operatorname{div}(L_{W_1}/L_{W_2})$, so $2[(W_1) - (W_2)] = 0$; hence $[(W_1) - (W_2)]$ corresponds to an element of $J_H[2](\overline{\mathbb{F}_q})$ (the two-torsion subgroup of J_H: that is, the kernel of multiplication by two). In particular, $[(W_1) - (W_2)] = [(W_2) - (W_1)]$, so the divisor class $[(W_1) - (W_2)]$ corresponds to the pair $\{W_1, W_2\}$ of Weierstrass points, and hence to the quadratic factor $L_{W_1} L_{W_2}$ of $\widetilde{F}$.

Proposition 1. [illegible] $\mathcal{G}$ [illegible] H [illegible] $\mathbb{F}_q$ [illegible] $J_H[2](\overline{\mathbb{F}_q})$ [illegible] $(\mathbb{Z}/2\mathbb{Z})^3$

[illegible] Let $\{\{W_1', W_1''\}, \{W_2', W_2''\}, \{W_3', W_3''\}, \{W_4', W_4''\}\}$ be a partition of the Weierstrass points of H into four disjoint pairs. Each pair $\{W_i', W_i''\}$ corresponds to the two-torsion divisor class $[(W_i') - (W_i'')]$ in $J_H[2](\overline{\mathbb{F}_q})$. We associate the subgroup $S := \langle [(W_i') - (W_i'')] : 1 \le i \le 4 \rangle$ to the partition. Observe that

$$\sum_{i=1}^{4} [(W_i') - (W_i'')] = \Big[\operatorname{div}\big(w / \prod_{i=1}^{4} L_{W_i''}\big)\Big] = 0;$$

this is the only relation on the classes $[(W_i') - (W_i'')]$, so $S \cong (\mathbb{Z}/2\mathbb{Z})^3$. The action of $\mathcal{G}$ on $J_H[2](\overline{\mathbb{F}_q})$ corresponds to its action on the Weierstrass points, so if the partition is $\mathcal{G}$-stable, then the subgroup S is $\mathcal{G}$-stable. □

[illegible] Requiring the pairs of points to be disjoint ensures that the associated subgroup is 2-Weil isotropic. This is necessary for the quotient by the subgroup to be an isogeny of principally polarized abelian varieties (see §9).

[illegible] By "an $\mathbb{F}_q$-rational subgroup of $J_H[2](\overline{\mathbb{F}_q})$ isomorphic to $(\mathbb{Z}/2\mathbb{Z})^3$", we mean a $\mathcal{G}$-stable subgroup that is isomorphic to $(\mathbb{Z}/2\mathbb{Z})^3$ over $\overline{\mathbb{F}_q}$. We emphasize that the elements of the subgroup need [illegible] be $\mathbb{F}_q$-rational themselves.

Definition 1. [illegible] H [illegible] tractable subgroups [illegible] $\mathcal{S}(H)$ [illegible] $\mathbb{F}_q$ [illegible] $J_H[2](\overline{\mathbb{F}_q})$

[illegible] Not every subgroup of $J_H[2](\overline{\mathbb{F}_q})$ that is the kernel of an isogeny of Jacobians is a tractable subgroup. For example, if $W_1, \ldots, W_8$ are the Weierstrass points of H, then the subgroup

$$\big\langle [(W_1) - (W_i) + (W_j) - (W_k)] : (i, j, k) \in \{(2,3,4), (2,5,6), (3,5,7)\} \big\rangle$$

is maximally 2-Weil isotropic, and hence is the kernel of an isogeny of Jacobians (see §9). However, this subgroup contains no nontrivial differences of Weierstrass points, and so cannot be a tractable subgroup.

Computing $\mathcal{S}(H)$ is straightforward if we identify each tractable subgroup with its corresponding partition of Weierstrass points. Each pair $\{W_i', W_i''\}$ of Weierstrass points corresponds to a quadratic factor of $\widetilde{F}$. Since the pairs are disjoint, the corresponding quadratic factors are pairwise coprime, and hence form (up scalar multiples) a factorization of the hyperelliptic polynomial $\widetilde{F}$. We therefore have a correspondence of tractable subgroups, partitions of Weierstrass points into pairs, and sets of quadratic polynomials (up to scalar multiples):

$$S \longleftrightarrow \big\{\{W_i', W_i''\} : 1 \le i \le 4\big\} \longleftrightarrow \big\{F_1, F_2, F_3, F_4\big\}, \text{ where } \widetilde{F} = F_1F_2F_3F_4.$$

Since the action of $\mathcal{G}$ on $J_H[2](\overline{\mathbb{F}_q})$ corresponds to its action on the set of Weierstrass points, the action of $\mathcal{G}$ on a tractable subgroup S corresponds to the action of $\mathcal{G}$ on the corresponding set $\{F_1, F_2, F_3, F_4\}$. In particular, S is $\mathbb{F}_q$-rational precisely when $\{F_1, F_2, F_3, F_4\}$ is fixed by $\mathcal{G}$. The factors F_i are themselves defined over $\mathbb{F}_q$ precisely when the corresponding points of S are $\mathbb{F}_q$-rational.

We can use this information to compute $\mathcal{S}(H)$. The set of pairs of Weierstrass points contains a $\mathcal{G}$-orbit $\big(\{W_{i_1}', W_{i_1}''\}, \ldots, \{W_{i_n}', W_{i_n}''\}\big)$ if and only if (possibly after exchanging some of the W_{i_k}' with the W_{i_k}'') either both $(W_{i_1}', \ldots, W_{i_n}')$ and $(W_{i_1}'', \ldots, W_{i_n}'')$ are $\mathcal{G}$-orbits or $(W_{i_1}', \ldots, W_{i_n}', W_{i_1}'', \ldots, W_{i_n}'')$ is a $\mathcal{G}$-orbit. Every $\mathcal{G}$-orbit of Weierstrass points corresponds to an $\mathbb{F}_q$-irreducible factor of F. Elementary calculations therefore yield the following useful lemma, as well as algorithms to compute all of the $\mathbb{F}_q$-rational tractable subgroups of $J_H[2](\overline{\mathbb{F}_q})$.

Lemma 1. [illegible] $H : w^2 = \widetilde{F}(u, v)$ [illegible] 3 [illegible] $\mathbb{F}_q$ [illegible] $\mathcal{S}(H)$ [illegible] $\mathbb{F}_q$ [illegible] $\widetilde{F}$ [illegible]

[illegible] $\mathbb{F}_q$ [illegible] $\widetilde{F}$	$\#\mathcal{S}(H)$
$(8), (6,2), (6,1,1), (4,2,1,1)$	1
$(4,4)$	5
$(4,2,2), (4,1,1,1,1), (3,3,2), (3,3,1,1)$	3
$(2,2,2,1,1)$	7
$(2,2,1,1,1,1)$	9
$(2,1,1,1,1,1,1)$	15
$(2,2,2,2)$	25
$(1,1,1,1,1,1,1,1)$	105
[illegible]	0

4 The Trigonal Construction

We will now briefly outline the theoretical aspects of constructing isogenies with tractable kernels. We will make the construction completely explicit in §5 and §6.

Definition 2. [illegible] $S = \langle[(W_i') - (W_i'')] : 1 \le i \le 4\rangle$ [illegible] $g : \mathbb{P}^1 \to \mathbb{P}^1$ [illegible] trigonal map for S [illegible] g [illegible] 3 [illegible] $g(\pi(W_i')) = g(\pi(W_i''))$ [illegible] $1 \le i \le 4$

Given a trigonal map g, Recillas' trigonal construction [12] specifies a curve X of genus 3 and a map $f : X \to \mathbb{P}^1$ of degree 4. The isomorphism class of X is independent of the choice of g. Theorem 1, due to Donagi and Livné, states that if g is a trigonal map for S, then S is the kernel of an isogeny from J_H to J_X.

Theorem 1 (Donagi and Livné [5, §5]). S … $J_H[2](\overline{\mathbb{F}_q})$ … $g : \mathbb{P}^1 \to \mathbb{P}^1$ … S … X … g … $\phi : J_H \to J_X$ … $\overline{\mathbb{F}_q}$ … S

We will give only a brief description of the geometry of X here, concentrating instead on its explicit construction; we refer the reader to Recillas [12], Donagi [4, §2], Birkenhake and Lange [1, §12.7], and Vakil [15] for the geometrical theory (and proofs). The isogeny is analogous to the well-known Richelot isogeny in genus 2 (see Bost and Mestre [2] and Donagi and Livné [5]).

In abstract terms, if U is the subset of the codomain of g above which $g \circ \pi$ is unramified, then X is by definition the closure of the curve over U representing the pushforward to U of the sheaf of sections of $\pi : (g \circ \pi)^{-1}(U) \to g^{-1}(U)$ (in the étale topology). This means in particular that the $\overline{\mathbb{F}_q}$-points of X over an $\overline{\mathbb{F}_q}$-point P of U represent partitions of the six $\overline{\mathbb{F}_q}$-points of $(g \circ \pi)^{-1}(P)$ into two sets of three exchanged by the hyperelliptic involution. The fibre product of H and X over $\mathbb{P}^1$ (with respect to $g \circ \pi$ and f) is the union of two isomorphic curves, R and R', which are exchanged by the involution on $H \times_{\mathbb{P}^1} X$ induced by the hyperelliptic involution. The natural projections induce coverings $\pi_H : R \to H$ and $\pi_X : R \to X$ of degrees 2 and 3, respectively, so R is a $(3, 2)$-correspondence between H and X. The map $(\pi_X)_* \circ (\pi_H)^*$ on divisor classes (that is, pulling back from H to R, then pushing forward onto X) induces an isogeny $\phi : J_H \to J_X$ with kernel S.[1] If we replace R with R' in the above, we obtain an isogeny isomorphic to $-\phi$. Thus, up to sign, the construction of the isogeny depends only on the subgroup S. The curves and morphisms described above form the commutative diagrams shown in Fig. 1.

The hyperelliptic Jacobians form a codimension-1 subspace of the moduli space of 3-dimensional principally polarized abelian varieties. Naïvely, then, if X is a curve of genus 3 selected at random, then the probability that X is hyperelliptic is inversely proportional to q; for cryptographically relevant sizes of q, this probability should be negligible. This is consistent with our experimental observations. In the sequel, by "a randomly chosen curve H and subgroup S in $\mathcal{S}(H)$", we mean a randomly chosen hyperelliptic curve H (in the sense of §2), together with a subgroup S uniformly randomly chosen from $\mathcal{S}(H)$.

Hypothesis 1. … X … H … S … $\mathcal{S}(H)$ …

[1] Recall that $(\pi_H)\ (\sum_{P\ \ H} n_P(P)) = \sum_{P\ \ H} n_P \sum_{Q\ \ \pi_H^{-1}(P)}(Q)$, with appropriate multiplicities where π_H ramifies, and $(\pi_X)\ (\sum_{Q\ \ R} m_Q(Q)) = \sum_{Q\ \ R} m_Q(\pi_X(Q))$.

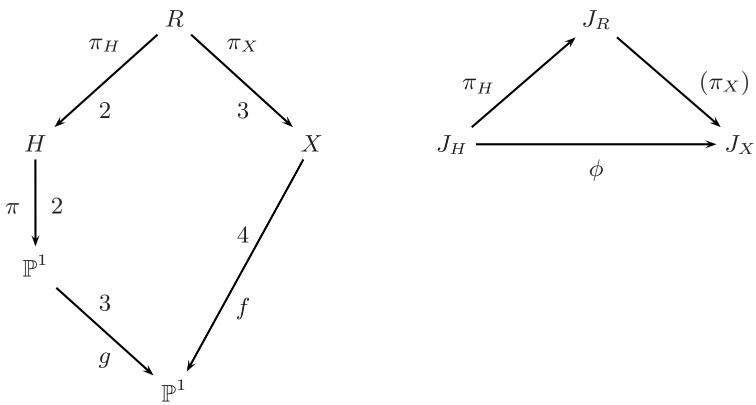


Fig. 1. The curves, Jacobians, and morphisms of §4

5 Computing Trigonal Maps

Suppose we are given a tractable subgroup S of $J_H[2](\overline{\mathbb{F}_q})$, corresponding to a partition $\{\{W_i', W_i''\} : 1 \leq i \leq 4\}$ of the Weierstrass points of H into pairs. In this section, we compute polynomials $N(x) = x^3 + ax + b$ and $D(x) = x^2 + cx + d$ such that the rational map $g : x \mapsto t = N(x)/D(x)$ defines a trigonal map for S. Choosing N and D to have degrees 3 and 2 respectively ensures that g maps the point at infinity to the point at infinity; this will be useful to us in §6.

By definition, $g : \mathbb{P}^1 \to \mathbb{P}^1$ is a degree-3 map with $g(\pi(W_i')) = g(\pi(W_i''))$ for $1 \leq i \leq 4$. We will express g as a composition of maps $g = p \circ e$, where $e : \mathbb{P}^1 \to \mathbb{P}^3$ is the rational normal embedding defined by

$$e : (u : v) \longmapsto (u_0 : u_1 : u_2 : u_3) = (u^3 : u^2 v : uv^2 : v^3),$$

and $p : \mathbb{P}^3 \to \mathbb{P}^1$ is the projection defined as follows. For each $1 \leq i \leq 4$, we let L_i denote the line in $\mathbb{P}^3$ passing through $e(\pi(W_i'))$ and $e(\pi(W_i''))$. There exists at least one line L intersecting all four of the L_i (generically, there are two). We take p to be the projection away from L; then $p(e(\pi(W_i'))) = p(e(\pi(W_i'')))$ for $1 \leq i \leq 4$, so $g = p \circ e$ is a trigonal map for S. Given equations for L, we can use linear algebra to compute a, b, c, and d in $\mathbb{F}_q$ such that

$$L = V(u_0 + au_2 + bu_3, u_1 + cu_2 + du_3) .$$

The projection $p : \mathbb{P}^3 \to \mathbb{P}^1$ away from L is then defined by

$$p : (u_0 : u_1 : u_2 : u_3) \longmapsto (u_0 + au_2 + bu_3 : u_1 + cu_2 + du_3),$$

and therefore $g = p \circ e$ is defined by

$$g : (u : v) \longmapsto (u^3 + auv^2 + bv^3 : u^2 v + cuv^2 + dv^3).$$

Therefore, if we set $N(x) = x^3 + ax + b$ and $D(x) = x^2 + cx + d$, then g will be defined by the rational map $x \longmapsto N(x)/D(x)$.

To compute equations for L, we will use the classical theory of [illegible] (see Griffiths and Harris [7, §1.5] for details). The set of lines in $\mathbb{P}^3$ has the structure of an algebraic variety $\mathrm{Gr}(1,3)$, called the Grassmannian. There is a convenient model for $\mathrm{Gr}(1,3)$ as a quadric hypersurface in $\mathbb{P}^5$: if $v_0, \ldots, v_5$ are coordinates on $\mathbb{P}^5$, then we may take

$$\mathrm{Gr}(1,3) := V(v_0v_3 + v_1v_4 + v_2v_5)\,.$$

Lemma 2. [illegible] $\mathrm{Gr}(1,3)(\overline{\mathbb{F}_q})$ [illegible] $\mathbb{P}^3$ [illegible]

[illegible] $\mathrm{Gr}(1,3)(\overline{\mathbb{F}_q})$ [illegible] $(p_0 : p_1 : p_2 : p_3)$ [illegible] $(q_0 : q_1 : q_2 : q_3)$ [illegible] $\mathbb{P}^3$ [illegible]

$$\left(\begin{vmatrix} p_0 & p_1 \\ q_0 & q_1 \end{vmatrix} : \begin{vmatrix} p_0 & p_2 \\ q_0 & q_2 \end{vmatrix} : \begin{vmatrix} p_0 & p_3 \\ q_0 & q_3 \end{vmatrix} : \begin{vmatrix} p_2 & p_3 \\ q_2 & q_3 \end{vmatrix} : \begin{vmatrix} p_3 & p_1 \\ q_3 & q_1 \end{vmatrix} : \begin{vmatrix} p_1 & p_2 \\ q_1 & q_2 \end{vmatrix} \right).$$

[illegible] $\mathbb{P}^3$ [illegible] $(\gamma_0 : \cdots : \gamma_5)$ [illegible] $\mathrm{Gr}(1,3)(\overline{\mathbb{F}_q})$ [illegible]

$$V\begin{pmatrix} 0u_0 - \gamma_3u_1 - \gamma_4u_2 - \gamma_5u_3, \\ \gamma_3u_0 + \ 0u_1 - \gamma_2u_2 + \gamma_1u_3, \\ \gamma_4u_0 + \gamma_2u_1 + \ 0u_2 - \gamma_0u_3, \\ \gamma_5u_0 - \gamma_1u_1 + \gamma_0u_2 + \ \ 0u_3 \end{pmatrix}$$

([illegible])

[illegible] $(\gamma_0 : \cdots : \gamma_5)$ [illegible] $\mathrm{Gr}(1,3)(\overline{\mathbb{F}_q})$ [illegible] L, [illegible] $\mathrm{Gr}(1,3)(\overline{\mathbb{F}_q})$ [illegible] L [illegible] $\sum_{i=0}^{5} \gamma_i v_{i+3}$, [illegible] 6

Assume that S is represented by a set $\{F_i = a_iu^2 + b_iuv + c_iv^2 : 1 \leq i \leq 4\}$ of quadratics, with each F_i corresponding to the pair $\{W_i', W_i''\}$ of Weierstrass points. Elementary calculations show that the point on $\mathrm{Gr}(1,3)$ corresponding to the line L_i through $e(\pi(W_i'))$ and $e(\pi(W_i''))$ has coordinates

$$(c_i^2 : -c_ib_i : b_i^2 - a_ic_i : a_i^2 : a_ib_i : a_ic_i).$$

If $(\gamma_0 : \cdots : \gamma_5)$ is a point in $\mathrm{Gr}(1,3)(\overline{\mathbb{F}_q})$ corresponding to a candidate for L, then by the second part of Lemma 2 we have $M(\gamma_0, \ldots, \gamma_5)^T = 0$, where

$$M = \begin{pmatrix} a_1^2 & a_1b_1 & a_1c_1 & c_1^2 & -c_1b_1 & (b_1^2 - a_1c_1) \\ a_2^2 & a_2b_2 & a_2c_2 & c_2^2 & -c_2b_2 & (b_2^2 - a_2c_2) \\ a_3^2 & a_3b_3 & a_3c_3 & c_3^2 & -c_3b_3 & (b_3^2 - a_3c_3) \\ a_4^2 & a_4b_4 & a_4c_4 & c_4^2 & -c_4b_4 & (b_4^2 - a_4c_4) \end{pmatrix}. \tag{1}$$

The kernel of M is two-dimensional, corresponding to a line in $\mathbb{P}^5$. Let $\{\underline{\alpha}, \underline{\beta}\}$ be a basis for $\ker M$, writing $\underline{\alpha} = (\alpha_0, \ldots, \alpha_5)$ and $\underline{\beta} = (\beta_0, \ldots, \beta_5)$. If S is $\mathbb{F}_q$-rational, then so is $\ker M$, so we may take the α_i and β_i to be in $\mathbb{F}_q$. We want to find a point $P_L = (\alpha_0 + \lambda\beta_0 : \cdots : \alpha_5 + \lambda\beta_5)$ where the line in $\mathbb{P}^5$ corresponding to $\ker M$ intersects with $\mathrm{Gr}(1,3)$. The points $(u_0 : \ldots : u_3)$ on the line L in $\mathbb{P}^3$ corresponding to P_L satisfy $(M_{\underline{\alpha}} + \lambda M_{\underline{\beta}})(u_0, \ldots, u_3)^T = 0$, where

$$M_{\underline{\alpha}} := \begin{pmatrix} 0 & -\alpha_3 & -\alpha_4 & -\alpha_5 \\ \alpha_3 & 0 & -\alpha_2 & \alpha_1 \\ \alpha_4 & \alpha_2 & 0 & -\alpha_0 \\ \alpha_5 & -\alpha_1 & \alpha_0 & 0 \end{pmatrix} \quad \text{and} \quad M_{\underline{\beta}} := \begin{pmatrix} 0 & -\beta_3 & -\beta_4 & -\beta_5 \\ \beta_3 & 0 & -\beta_2 & \beta_1 \\ \beta_4 & \beta_2 & 0 & -\beta_0 \\ \beta_5 & -\beta_1 & \beta_0 & 0 \end{pmatrix}.$$

By part (2) of Lemma 2, the rank of $M_{\underline{\alpha}} + \lambda M_{\underline{\beta}}$ is 2. Using the expression

$$\det(M_{\underline{\alpha}} + \lambda M_{\underline{\beta}}) = \Big(\frac{1}{2}\big(\sum_{i=0}^{6} \beta_i\beta_{i+3}\big)\lambda^2 + \big(\sum_{i=0}^{6} \alpha_i\beta_{i+3}\big)\lambda + \frac{1}{2}\sum_{i=0}^{6} \alpha_i\alpha_{i+3}\Big)^2 \qquad (2)$$

(where the subscripts are taken modulo 6), we see that this occurs precisely when $\det(M_{\underline{\alpha}} + \lambda M_{\underline{\beta}}) = 0$. We can therefore solve $\det(M_{\underline{\alpha}} + \lambda M_{\underline{\beta}}) = 0$ to determine a value for λ, and to see that $\mathbb{F}_q(\lambda)$ is at most a quadratic extension of $\mathbb{F}_q$. Considering the discriminant of $\det(M_{\underline{\alpha}} + \lambda M_{\underline{\beta}})$ gives us an explicit criterion for determining whether a given tractable subgroup has a rational trigonal map.

Proposition 2. S $\mathcal{S}(H)$ $\{\underline{\alpha} = (\alpha_i), \underline{\beta} = (\beta_i)\}$ $\mathbb{F}_q$ M (1) $\mathbb{F}_q$ S

$$\Big(\sum_{i=0}^{6} \alpha_i\beta_{i+3}\Big)^2 - \Big(\sum_{i=0}^{6} \alpha_i\alpha_{i+3}\Big)\Big(\sum_{i=0}^{6} \beta_i\beta_{i+3}\Big)$$

$\mathbb{F}_q$ 6

Finally, we use Gaussian elimination to compute a, b, c, and d in $\mathbb{F}_q(\lambda)$ such that $(1, 0, a, b)$ and $(0, 1, c, d)$ generate the rowspace of $M_{\underline{\alpha}} + \lambda M_{\underline{\beta}}$. We may then take $L = V(u_0 + au_2 + u_3, u_1 + cu_2 + du_3)$. Both L and the projection $p : \mathbb{P}^3 \to \mathbb{P}^1$ with centre L are defined over $\mathbb{F}_q(\lambda)$. Having computed L, we compute the projection p, the embedding e, and the trigonal map $g = p \circ e$ as above.

Proposition 2 shows that the rationality of a trigonal map for a tractable subgroup S depends only upon whether an element of $\mathbb{F}_q$ depending on S is a square. It seems reasonable to assume that these field elements are uniformly distributed for random choices of H and S, and indeed this is consistent with our experimental observations. Since a uniformly randomly chosen element of $\mathbb{F}_q$ is a square with probability $\sim 1/2$, we propose the following hypothesis.

Hypothesis 2. $\mathbb{F}_q$ H $\mathbb{F}_q$ S $\mathcal{S}(H)$ $1/2$

6 Equations for the Isogeny

Suppose we have a tractable subgroup S and a trigonal map g for S. We will now perform an explicit trigonal construction on g to compute a curve X and an isogeny $\phi : J_H \to J_X$ with kernel S. We assume that g has been derived as in §5, and in particular that g maps the point at infinity to the point at infinity.

Let U be the subset of $\mathbb{A}^1 = \mathbb{P}^1 \setminus \{(1:0)\}$ above which $g \circ \pi$ is unramified. We let $X|_U$ denote $f^{-1}(U)$, and let $H|_U$ denote $(g \circ \pi)^{-1}(U)$.

By definition, every point P in $X|_U(\overline{\mathbb{F}_q})$ corresponds to a pair of triples of points in $H|_U(\overline{\mathbb{F}_q})$, exchanged by the hyperelliptic involution, with each triple supported on the fibre of $g \circ \pi$ over $f(P)$. We will construct a model of the abstract curve $X|_U$ in $U \times \mathbb{A}^6$. We will not prove that our model is isomorphic to the abstract curve, but we will exhibit a bijection of geometric points.

To be more explicit, suppose Q is a generic point of U. Since $g \circ \pi$ is unramified above Q, we may choose preimages P_1, P_2 and P_3 of Q such that

$$(g \circ \pi)^{-1}(Q) = \{P_1, P_2, P_3, \iota(P_1), \iota(P_2), \iota(P_3)\}. \tag{3}$$

The four points on X in the preimage $f^{-1}(Q)$ correspond to partitions of the six points in $(g \circ \pi)^{-1}(Q)$ into two unordered triples exchanged by the hyperelliptic involution:

$$f^{-1}(Q) = \left\{ \begin{array}{l} Q_1 \leftrightarrow \big\{\{P_1, P_2, P_3\},\ \{\iota(P_1), \iota(P_2), \iota(P_3)\}\big\}, \\ Q_2 \leftrightarrow \big\{\{P_1, \iota(P_2), \iota(P_3)\},\ \{\iota(P_1), P_2, P_3\}\big\}, \\ Q_3 \leftrightarrow \big\{\{\iota(P_1), P_2, \iota(P_3)\},\ \{P_1, \iota(P_2), P_3\}\big\}, \\ Q_4 \leftrightarrow \big\{\{\iota(P_1), \iota(P_2), P_3\},\ \{P_1, P_2, \iota(P_3)\}\big\} \end{array} \right\}. \tag{4}$$

Every triple is cut out by an ideal $(a(x), y - b(x))$, where a is a cubic polynomial, b is a quadratic polynomial, and $b^2 \equiv F \pmod{a}$. If we require a to be monic, then there is a one-to-one correspondence between such ideals and triples; this is the well-known [illegible]. The triple is defined over $\mathbb{F}_q$ if and only if a and b are defined over $\mathbb{F}_q$. For example, the triple $\{P_1, P_2, P_3\}$ corresponds to the ideal $(a(x), y - b(x))$ where $a(x) = \prod_i (x - x(P_i))$ and b satisfies $y(P_i) = b(x(P_i))$ for $1 \leq i \leq 3$; the Lagrange interpolation formula may be used to compute b. If $(a(x), y - b(x))$ corresponds to one triple in a partition, then $(a(x), y + b(x))$ corresponds to the other triple. The union of the triples equals the whole fibre $(g \circ \pi)^{-1}(Q)$, and since the union of the triples is cut out by the product of the corresponding ideals, we know that $a(x)$ must cut out the fibre of $g \circ \pi$ over Q. Therefore, we have $a(x) = N(x) - t(Q)D(x)$.

For notational convenience, we define

$$G(t, x) = x^3 + g_2(t)x^2 + g_1(t)x + g_0(t) := N(x) - tD(x).$$

Let f_0, f_1, and f_2 be the elements of $\mathbb{F}_q[t]$ such that

$$f_0(t) + f_1(t)x + f_2(t)x^2 \equiv F(x) \pmod{G(t, x)}.$$

The triples in the pairs over the generic point of U have Mumford representatives of the form $(G(t, x), y - (b_0 + b_1 x + b_2 x^2))$, where

$$(b_0 + b_1 x + b_2 x^2)^2 \equiv F(x) \pmod{G(t, x)}. \tag{5}$$

Viewing b_0, b_1, and b_2 as coordinates on $\mathbb{A}^3$, we expand both sides of (5) modulo $G(t,x)$ and equate coefficients to obtain a variety $\widetilde{X}$ in $U \times \mathbb{A}^3$ parametrizing triples:

$$\widetilde{X} := V(c_0(t,b_0,b_1,b_2), c_1(t,b_0,b_1,b_2), c_2(t,b_0,b_1,b_2)),$$

where

$$\begin{aligned} c_0(t,b_0,b_1,b_2) &= g_2(t)g_0(t)b_2^2 - 2g_0(t)b_2b_1 + b_0^2 - f_0(t), \\ c_1(t,b_0,b_1,b_2) &= (g_2(t)g_1(t) - g_0(t))b_2^2 - 2g_1(t)b_2b_1 + 2b_1b_0 - f_1(t), \text{ and} \\ c_2(t,b_0,b_1,b_2) &= (g_2(t)^2 - g_1(t))b_2^2 - 2g_2(t)b_2b_1 + 2b_2b_0 + b_1^2 - f_2(t). \end{aligned} \tag{6}$$

The Mumford representatives corresponding to the triples in each pair are exchanged by the involution $\iota_* : \widetilde{X} \longrightarrow \widetilde{X}$ defined by

$$\iota_* : (t, b_0, b_1, b_2) \longmapsto (t, -b_0, -b_1, -b_2);$$

the curve $X|_U$ is therefore the quotient of $\widetilde{X}$ by the involution ι_*. To form this quotient, let $m : U \times \mathbb{A}^3 \longrightarrow U \times \mathbb{A}^6$ be the map defined by

$$m : (t, b_0, b_1, b_2) \longmapsto (t, b_{00}, b_{01}, b_{02}, b_{11}, b_{12}, b_{22}) = (t, b_0^2, b_0b_1, b_0b_2, b_1^2, b_1b_2, b_2^2);$$

the image B of m is the variety defined by

$$B = V\begin{pmatrix} b_{01}^2 - b_{00}b_{11},\ b_{01}b_{02} - b_{00}b_{12},\ b_{02}^2 - b_{00}b_{22}, \\ b_{02}b_{11} - b_{01}b_{12},\ b_{02}b_{12} - b_{01}b_{22},\ b_{12}^2 - b_{11}b_{22} \end{pmatrix} \subset U \times \mathbb{A}^3.$$

We have $X|_U = m(\widetilde{X})$, so

$$X|_U = V\begin{pmatrix} g_2g_0b_{22} - 2g_0b_{12} + b_{00} - f_0, \\ (g_2g_1 - g_0)b_{22} - 2g_1b_{12} + 2b_{01} - f_1, \\ (g_2^2 - g_1)b_{22} - 2g_2b_{12} + 2b_{02} + b_{11} - f_2 \end{pmatrix} \cap B \subset U \times \mathbb{A}^6. \tag{7}$$

Consider again the fibre of $f : X \to \mathbb{P}^1$ over the generic point $Q = (t)$ of U (as in (4)). If $\{P_1, P_2, P_3\}$ is one of the triples in a pair in the fibre, then by the Lagrange interpolation formula the value of b_2 at the corresponding point of $\widetilde{X}$ is

$$b_2 = \sum y(P_i)/((x(P_i) - x(P_j))(x(P_i) - x(P_k))),$$

where the sum is taken over the cyclic permutations (i,j,k) of $(1,2,3)$. Interpolating for all triples in the pairs in the fibre, an elementary but involved symbolic calculation shows that if we define Δ_1, Δ_2, and Δ_3 by

$$\Delta_i := (x(P_j) - x(P_k))^2$$

and Γ_1, Γ_2, and Γ_3 by

$$\Gamma_i := \left(f_2(t)x(P_i)^2 + f_1(t)x(P_i) + f_0(t)\right)\Delta_i = F(x(P_i))\Delta_i$$

for each cyclic permutation (i,j,k) of $(1,2,3)$, and set

$$\Delta := \Delta_1\Delta_2\Delta_3,$$

then b_2 satisfies

$$\Big(\Delta b_2^4 - 2\big(\sum_i \Gamma_i\big)b_2^2 + \frac{1}{\Delta}\Big(2\big(\sum_i \Gamma_i^2\big) - \big(\sum_i \Gamma_i\big)^2\Big)\Big)^2 - 64\big(\prod_i \Gamma_i\big)b_2^2 = 0. \quad (8)$$

Now Δ, $\sum_i \Gamma_i$, $\sum_i \Gamma_i^2$, and $\prod_i \Gamma_i$ are symmetric functions with respect to permutations of the points in the fibre $g^{-1}(Q) = g^{-1}((t))$. They are therefore polynomials in the homogeneous elementary symmetric functions

$$e_1 = \sum x(P_i), \quad e_2 = \sum x(P_i)x(P_j), \quad \text{and} \quad e_3 = \prod x(P_i),$$

which are polynomials in t. Indeed, the e_i are given by the coefficients of $G(t,x)$:

$$e_1 = -g_2(t), \quad e_2 = g_1(t), \quad \text{and} \quad e_3 = -g_0(t).$$

Expressing Δ, $\sum_i \Gamma_i$, $\sum_i \Gamma_i^2$, and $\prod_i \Gamma_i$ in terms of f_0, f_1, f_2, g_0, g_1, and g_2, and then simplifying, we define δ_4, δ_2, and δ_0 by

$$\begin{aligned}
\delta_4 &:= -27g_0^2 + 18g_0g_1g_2 - 4g_0g_2^3 - 4g_1^3 + g_1^2g_2^2,\\
\delta_2 &:= 12f_0g_1 - 4f_0g_2^2 - 18f_1g_0 + 2f_1g_1g_2 + 12f_2g_0g_2 - 4f_2g_1^2,\\
\delta_0 &:= -4f_0f_2 + f_1^2,
\end{aligned}$$

and s by

$$\begin{aligned}
s := {} & f_0^3 - f_0^2f_1g_2 - 2f_0^2f_2g_1 + f_0^2f_2g_2^2 + f_0f_1^2g_1 + 3f_0f_1f_2g_0 - f_0f_1f_2g_1g_2 \\
& - 2f_0f_2^2g_0g_2 + f_0f_2^2g_1^2 - f_1^3g_0 + f_1^2f_2g_0g_2 - f_1f_2^2g_0g_1 + f_2^3g_0^2.
\end{aligned} \quad (9)$$

Since $s(t) = F(x(P_1))F(x(P_2))F(x(P_3)) = (y(P_1)y(P_2)y(P_3))^2$, there is a square root of $s(t)$ in $\overline{\mathbb{F}_q}[t]$; in fact, it is defined over $\mathbb{F}_q(\sqrt{s(0)})$. We therefore define

$$\delta_1 := 8\sqrt{s}. \quad (10)$$

With this notation (8) becomes $\big(\delta_4(t)b_2^4 + \delta_2(t)b_2^2 + \delta_0(t)\big)^2 - \delta_1(t)^2b_2^2 = 0$, and hence on $X|_U$ we have

$$\big(\delta_4(t)b_{22}^2 + \delta_2(t)b_{22} + \delta_0(t)\big)^2 - \delta_1(t)^2b_{22} = 0. \quad (11)$$

Observe that (11) gives us a (singular) affine plane model for X. We can also use (11) to compute a square root for b_{22} on $X|_U$: we have

$$b_{22} = \rho^2, \quad \text{where} \quad \rho := \frac{\delta_4(t)b_{22}^2 + \delta_2(t)b_{22} + \delta_0(t)}{\delta_1(t)}. \quad (12)$$

Given a point $(t, b_{00}, \ldots, b_{22})$ of $X|_U$, the two triples of points corresponding to the two points of $\widetilde{X}$ over $(t, b_{00}, \ldots, b_{22})$ have Mumford representatives

$$\Big(G(t,x), y - \big(\frac{b_{02}}{\rho} + \frac{b_{12}}{\rho}x + \frac{b_{22}}{\rho}x^2\big)\Big) \text{ and } \Big(G(t,x), y + \big(\frac{b_{02}}{\rho} + \frac{b_{12}}{\rho}x + \frac{b_{22}}{\rho}x^2\big)\Big). \quad (13)$$

We will now compute the Recillas correspondence R inducing the isogeny from J_H to J_X. We know that R is a component of the fibre product $H \times_{\mathbb{P}^1} X$

(with respect to $g \circ \pi$ and f). We may realise the open affine subset $H|_U \times_U X|_U$ as the subvariety $V(G(t,x))$ of $H|_U \times X|_U$. Now, $V(G(t,x))$ decomposes into two components: clearing denominators in (13), we find $V(G(t,x)) = R \cup R'$, where

$$R = V\big(G(t,x), (\delta_4(t)b_{22}^2 + \delta_2(t)b_{22} + \delta_0(t))y - \delta_1(t)(b_{02} + b_{12}x + b_{22}x^2)\big)$$

and

$$R' = V\big(G(t,x), (\delta_4(t)b_{22}^2 + \delta_2(t)b_{22} + \delta_0(t))y + \delta_1(t)(b_{02} + b_{12}x + b_{22}x^2)\big)\,.$$

The natural projections $\pi_X : R \to X$ and $\pi_H : R \to H$ send $(x, y, t, b_{00}, \ldots, b_{22})$ to $(t, b_{00}, \ldots, b_{22})$ and (x, y), respectively. On the level of divisor classes, the isogeny $\phi : J_H \to J_X$ is made explicit by the map

$$\phi = (\pi_X)_* \circ (\pi_H)^*.$$

In terms of ideals cutting out effective divisors, ϕ is realized by the map

$$I_D \longmapsto \left(I_D + \left(G(t,x), y - (\frac{b_{02}}{\rho} + \frac{b_{12}}{\rho}x + \frac{b_{22}}{\rho}x^2)\right)\right) \cap \mathbb{F}_q[s, t, b_{00}, \ldots, b_{22}].$$

Taking R' in place of R in the above gives an isogeny equal to $-\phi$.

It remains to determine the rationality of the isogeny. We see from (7) that X is defined over the field of definition of g. The correspondence R, and the isogeny ϕ, are both defined over the field of definition of ρ, which is $\mathbb{F}_q(\sqrt{s(0)})$. This gives us a useful criterion for when an $\mathbb{F}_q$-rational subgroup S and trigonal map g lead to an $\mathbb{F}_q$-rational isogeny.

Proposition 3. [illegible] S [illegible] $\mathcal{S}(H)$ [illegible] $\mathbb{F}_q$ [illegible] g [illegible] g [illegible] $\mathbb{F}_q$ [illegible] $s(0)$ [illegible] $\mathbb{F}_q$ [illegible] s [illegible] (9)

[illegible] If ϕ is not $\mathbb{F}_q$-rational, then J_X is a quadratic twist of J_H/S (see §9).

If we assume that the values $s(0)$ are uniformly distributed for randomly chosen H, S, and g, then the probability that $s(0)$ is a square in $\mathbb{F}_q$ is $1/2$. Indeed, it is easily seen that $s(0)$ is a square for H if and only if it is not a square for the quadratic twist of H. This suggests that the probability that we can compute an $\mathbb{F}_q$-rational ϕ given an $\mathbb{F}_q$-rational g for a randomly chosen H and S in $\mathcal{S}(H)$ is $1/2$. This is consistent with our experimental observations, so we propose Hypothesis 3.

Hypothesis 3. [illegible] H [illegible] $\mathbb{F}_q$ [illegible] S [illegible] $\mathcal{S}(H)$ [illegible] $\mathbb{F}_q$ [illegible] g [illegible] $\mathbb{F}_q$ [illegible] ϕ [illegible] S [illegible] $1/2$

7 Computing Isogenies

Suppose we are given a hyperelliptic curve H of genus 3, defined over $\mathbb{F}_q$, and a DLP in $J_H(\mathbb{F}_q)$ to solve. Our goal is to compute a nonsingular plane quartic curve C and an isogeny $J_H \to J_C$ so that we can reduce to a DLP in $J_C(\mathbb{F}_q)$.

We begin by computing the set $\mathcal{S}(H)$ of $\mathbb{F}_q$-rational tractable subgroups of $J_H[2](\overline{\mathbb{F}_q})$. For each S in $\mathcal{S}(H)$, we apply Proposition 2 to determine whether there exists an $\mathbb{F}_q$-rational trigonal map g for S. If so, we use the formulae of §5 to compute g; if not, we move on to the next S. Having computed g, we apply Proposition 3 to determine whether we can compute an isogeny over $\mathbb{F}_q$. If so, we use the formulae of §6 to compute equations for X and the isogeny $J_H \to J_X$; if not, we move on to the next S.

The formulae of §6 give an affine model of X in $\mathbb{A}^1 \times \mathbb{A}^6$. In order to apply Diem's algorithm to the DLP in J_X, we need a nonsingular plane quartic model of X: that is, a nonsingular curve $C \subset \mathbb{P}^2$ isomorphic to X, cut out by a quartic form. Such a model exists if and only if X is not hyperelliptic. To find C, we compute a basis $\mathcal{B}$ of the Riemann–Roch space of a canonical divisor of X. This is a routine geometrical calculation; some of the various approaches are listed in Hess [8]. In practice, the algorithms implemented in Magma [9] compute $\mathcal{B}$ very quickly. The three functions in $\mathcal{B}$ define a map $\psi : X \to \mathbb{P}^2$. If the image of ψ is a conic, then X is hyperelliptic; in this situation, we move on to the next S. Otherwise, the image of ψ is a nonsingular plane quartic C, and ψ restricts to an isomorphism $\psi : X \to C$.

If the procedure outlined above succeeds for some S in $\mathcal{S}(H)$, then we have computed an explicit $\mathbb{F}_q$-rational isogeny $\psi_* \circ \phi : J_H \to J_C$. We can then map our DLP from $J_H(\mathbb{F}_q)$ into $J_C(\mathbb{F}_q)$, and solve using Diem's algorithm.

We emphasize that the entire procedure is very fast: as we saw above, the curve X and the isogeny can be constructed using only low-degree polynomial arithmetic and low-dimensional linear algebra. For a rough idea of the computational effort involved, given a random H over a 160-bit prime field, a naïve implementation of our algorithms in Magma [9] computes the trigonal map g, the curve X, the nonsingular plane quartic C, and the isogeny $\phi : J_H \to J_C$ in a few seconds on a 1.2GHz laptop. Since the difficulty of the construction depends only upon the size of $\mathbb{F}_q$ (and [illegible] upon the size of the DLP subgroup of $J_H(\mathbb{F}_q)$), we may conclude that instances of the DLP in 160-bit Jacobians chosen for cryptography may also be reduced to instances of the DLP in non-hyperelliptic Jacobians in a mattter of seconds.

[illegible] We will give an example over a small field. Let H be the hyperelliptic curve over $\mathbb{F}_{37}$ defined by

$$H : y^2 = x^7 + 28x^6 + 15x^5 + 20x^4 + 33x^3 + 12x^2 + 29x + 2.$$

Using the ideas in §3, we see that J_H has one $\mathbb{F}_{37}$-rational tractable subgroup:

$$\mathcal{S}(H) = \{S\} \quad \text{where} \quad S = \left\{ \begin{array}{l} u^2 + \xi_1 uv + \xi_2 v^2,\ u^2 + \xi_1^{37} uv + \xi_2^{37} v^2, \\ u^2 + \xi_1^{37^2} uv + \xi_2^{37^2} v^2,\ uv + 20v^2 \end{array} \right\},$$

where ξ_1 is an element of $\mathbb{F}_{37^3}$ satisfying $\xi_1^3+29\xi_1^2+9\xi_1+13=0$, and $\xi_2=\xi_1^{50100}$. Applying the methods of §5, we compute polynomials

$$N(x)=x^3+16x+22 \quad \text{and} \quad D(x)=x^2+32x+18$$

such that $g : x \longmapsto N(x)/D(x)$ is an $\mathbb{F}_{37}$-rational trigonal map for S. Using the formulae of §6, we compute a curve $X \subset \mathbb{A}^1 \times \mathbb{A}^6$ of genus 3, defined by

$$X = V\begin{pmatrix} 19t^5+10t^4+12t^3+18t^2b_{22}+7t^2+36tb_{12}+15tb_{22}+t+b_{00}+30b_{12}+30, \\ 5t^5+26t^4+15t^3+32t^2b_{22}+23t^2+27tb_{12}+2tb_{22}+19t+2b_{01}+5b_{12}+15b_{22}+17, \\ 36t^5+29t^4+7t^3+t^2b_{22}+13t^2+2tb_{12}+32tb_{22}+21t+2b_{02}+b_{11}+21b_{22}+18, \\ b_{00}b_{11}-b_{01}^2, b_{00}b_{12}-b_{01}b_{02}, b_{00}b_{22}-b_{02}^2, b_{02}b_{11}-b_{01}b_{12}, b_{02}b_{12}-b_{01}b_{22}, b_{12}^2-b_{11}b_{22} \end{pmatrix}$$

together with a map on divisors inducing an isogeny from J_H to J_X with kernel S (we will not show the equations, for lack of space). Computing the canonical morphism of X, we find that X is non-hyperelliptic, and isomorphic to the nonsingular plane quartic curve

$$C = V\begin{pmatrix} u^4+26u^3v+2u^3w+17u^2v^2+9u^2vw+20u^2w^2+34uv^3+24uv^2w \\ +5uvw^2+36uw^3+19v^4+13v^3w+v^2w^2+23vw^3+5w^4 \end{pmatrix}.$$

Composing the isomorphism with the isogeny $J_H \to J_X$, we obtain an explicit isogeny $\phi : J_H \to J_C$. Using Magma, we can verify that J_H and J_C are isogenous by checking that the zeta functions of H and C are identical: indeed,

$$Z(H;T) = Z(C;T) = \frac{37^3T^6+4\cdot 37^2T^5-6\cdot 37T^4-240T^3-6T^2+4T+1}{37T^2-38T+1}.$$

If D and D' are the divisor classes on H with Mumford representatives $(x^2+13x+29, y-10x-2)$ and $(x^2+19x+18, y-15x-2)$, respectively, then $D'=[22359]D$. Applying ϕ, we find that

$$\phi(D) = [(7:18:1)+(34:34:1)-(18:22:1)-(15:33:1)] \quad \text{and}$$
$$\phi(D') = [(7:23:1)+(6:13:1)-(13:15:1)-(7:18:1)];$$

direct calculation verifies that $\phi(D')=[22359]\phi(D)$, as expected.

8 Expectation of Existence of Computable Isogenies

We conclude by estimating the proportion of genus 3 hyperelliptic Jacobians over $\mathbb{F}_q$ for which the methods of this article produce a rational isogeny — and thus the proportion of hyperelliptic curves for which the DLP may be solved using Diem's algorithm — as q tends to infinity. We will assume that if we are given a selection of $\mathbb{F}_q$-rational tractable subgroups, then it is equally probable that any one of them will yield a rational isogeny. This appears consistent with our experimental observations.

Hypothesis 4. If S_1 and S_2 are distinct subgroups in $\mathcal{S}(H)$, then the probability that we can compute an $\mathbb{F}_q$-rational isogeny with kernel S_1 is independent of the probability that we can compute an $\mathbb{F}_q$-rational isogeny with kernel S_2.

Theorem 2. ... $\mathcal{T}$ ... 8 ... T. $\mathcal{T}$ ... $\nu_T(n)$ ... n. T ... $s(T) = \#\mathcal{S}(H)$ ... H. any ... $\mathbb{F}_q$ ... $\mathbb{F}_q$ ... T ... q ... $J_H(\mathbb{F}_q)$ ... H ... 3 ... $\mathbb{F}_q$ ... $J_C(\mathbb{F}_q)$ ... C.

$$\sum_{T\in\mathcal{T}} \Big(\big(1-(1-1/4)^{s(T)}\big) / \prod_{n\in T} \big(\nu_T(n)! \cdot n^{\nu_T(n)}\big) \Big) \approx 0.1857. \tag{14}$$

Hypotheses 1, 2, 3, and 4 together imply that if H is a randomly chosen hyperelliptic curve of genus 3 over $\mathbb{F}_q$, then the probability that we will succeed in computing a rational isogeny from J_H is

$$1-(1-(1/2\cdot 1/2))^{\#\mathcal{S}(H)}. \tag{15}$$

Lemma 1 implies that $\mathcal{S}(H)$ depends only on the degrees of the irreducible factors of $\widetilde{F}$. For each T in $\mathcal{T}$, let $N_q(T)$ denote the number of homogeneous squarefree polynomials over $\mathbb{F}_q$ whose multiset of degrees of irreducible factors coincides with T. By (15), the expectation that we can compute an $\mathbb{F}_q$-rational isogeny from the Jacobian of a randomly chosen hyperelliptic curve to the Jacobian of a non-hyperelliptic curve using the methods in this article is

$$E_q := \frac{\sum_{T\in\mathcal{T}} (1-(1-1/4)^{s(T)}) N_q(T)}{\sum_{T\in\mathcal{T}} N_q(T)}.$$

Let $N_q(n)$ denote the number of monic irreducible polynomials of degree n over $\mathbb{F}_q$; clearly $N_q(T) = (q-1)\prod_{n\in T}\binom{N_q(n)}{\nu_T(n)}$. Computing $N_q(T)$ is a straightforward combinatorial exercise: we find that $N_q(n) = q^n/n + O(q^{n-1})$, so

$$N_q(T) = \Big(\prod_{n\in T} (\nu_T(n)!\cdot n^{\nu_T(n)})^{-1}\Big) q^9 + O(q^8),$$

and $\sum_{T\in\mathcal{T}} N_q(T) = q^9 + O(q^8)$. Therefore, as q tends to infinity, we have

$$\lim_{q\to\infty} E_q = \sum_{T\in\mathcal{T}} \Big((1-(1-1/4)^{s(T)}) / \prod_{n\in T} (\nu_T(n)!\cdot n^{\nu_T(n)}) \Big).$$

The result follows upon explicitly computing this sum using the values for $s(T)$ derived in Lemma 1. □

Theorem 2 gives the expectation that we can construct an explicit isogeny for a randomly selected hyperelliptic curve. However, looking at the table in Lemma 1, we see that we can ensure that a particular curve has no rational isogenies if its hyperelliptic polynomial has an irreducible factor of degree 5 or 7 (or a single irreducible factor of degree 3). It may be difficult to efficiently construct a curve in this form if we are using the CM construction, for example, to ensure that the Jacobian has a large prime-order subgroup. In any case, it is interesting to note that the security of genus 3 hyperelliptic Jacobians depends significantly upon the factorization of their hyperelliptic polynomials. This observation has no analogue for elliptic curves or Jacobians of genus 2 curves.

We noted in §4 that the isomorphism class of the curve X in the trigonal construction is independent of the choice of trigonal map. If there is no rational trigonal map for a given subgroup S, then the methods of §5 construct a pair of Galois-conjugate trigonal maps g_1 and g_2 (corresponding to the roots of (2)) instead. Applying the trigonal construction to g_1 and g_2, we obtain a pair of curves X_1 and X_2 over $\mathbb{F}_{q^2}$, which must be twists. If the isomorphism between these two curves was made explicit, then Galois descent could be used to compute a curve X in their isomorphism class defined over $\mathbb{F}_q$, and hence a nonsingular plane quartic C and isogeny $J_H \to J_C$ over $\mathbb{F}_q$. This approach would allow us to replace the 1/4 in (15) and (14) with 1/2, raising the expectation of success in Theorem 2 to over 30%.

9 Other Isogenies

In this article, we have used a special kind of $(2, 2, 2)$-isogeny for moving instances of the DLP from hyperelliptic to non-hyperelliptic Jacobians. More generally, we can consider using other types of isogenies. There are two important issues to consider here: the first is a theoretical restriction on the types of subgroups S of J_H that can be kernels of isogenies of Jacobians, and the second is a practical restriction on the isogenies that we can currently compute.

Suppose J_H is a hyperelliptic Jacobian, and S a (finite) $\mathbb{F}_q$-rational subgroup of J_H. The quotient $J_H \to J_H/S$ exists as an isogeny of abelian varieties (see Serre [14, §III.3.12], for example). For the quotient to be an isogeny of Jacobians, there must be an integer m such that S is a maximal isotropic subgroup with respect to the m-Weil pairing (see Proposition 16.8 of Milne [10]): this ensures that the canonical polarization on J_H induces a principal polarization on the quotient. The simplest such subgroups have the form $(\mathbb{Z}/l\mathbb{Z})^3$ where l is prime. The theorem of Oort and Ueno [11] then guarantees that there will be an isomorphism over $\overline{\mathbb{F}}_q$ from J_H/S to the Jacobian J_X of some (possibly reducible) curve X. Standard arguments from Galois cohomology (see Serre [13, §III.1], for example) show that the isomorphism is defined over either $\mathbb{F}_q$ or $\mathbb{F}_{q^2}$, so J_H/S is either isomorphic to J_X over $\mathbb{F}_q$ or a quadratic twist of J_X. We can expect X to be isomorphic to a non-hyperelliptic curve C. To compute an $\mathbb{F}_q$-rational isogeny from J_H to a non-hyperelliptic Jacobian, therefore, the minimum requirement is an $\mathbb{F}_q$-rational l-Weil isotropic subgroup of $J_H(\overline{\mathbb{F}_q})$ isomorphic to $(\mathbb{Z}/l\mathbb{Z})^3$ for some prime l.

The second and more serious problem is the lack of general constructions for isogenies in genus 3. Apart from integer and Frobenius endomorphisms, we know of no constructions for explicit isogenies of general Jacobians of genus 3 hyperelliptic curves other than the one presented here. This situation stands in marked contrast to the case of isogenies of elliptic curves, which have been made completely explicit by Vélu [16]. Deriving general formulae for explicit isogenies in genus 3 (and 2) remains a significant problem in computational number theory.

Acknowledgements

This work was supported by EPSRC grant EP/C014839/1, and a large part of it was completed in the Department of Mathematics at Royal Holloway, University of London. The author gratefully acknowledges Roger Oyono and Christophe Ritzenthaler for discussions which inspired this research, Steven Galbraith for his advice, and the (anonymous) referees for their suggestions.

References

1. Birkenhake, C., Lange, H.: Complex abelian varieties (2e). In: Grundlehren der mathematischen Wissenschaften, vol. 302, Springer, Heidelberg (2004)
2. Bost, J.-B., Mestre, J.-F.: Moyenne arithmético-géométrique et périodes des courbes de genre 1 et 2. Gaz. Math. Soc. France 38, 36–64 (1988)
3. Diem, C.: An index calculus algorithm for plane curves of small degree. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, Springer, Heidelberg (2006)
4. Donagi, R.: The fibres of the Prym map. Curves, Jacobians, and abelian varieties (Amherst, MA, 1990), Contemp. Math. 136, 55–125 (1992)
5. Donagi, R., Livné, R.: The arithmetic-geometric mean and isogenies for curves of higher genus. Ann. Scuola Norm. Sup. Pisa Cl. Sci(4) 28(2), 323–339 (1999)
6. Gaudry, P., Thomé, E., Thériault, N., Diem, C.: A double large prime variation for small genus hyperelliptic index calculus. Math. Comp. 76, 475–492 (2007)
7. Griffiths, P., Harris, J.: Principles of Algebraic Geometry. Wiley and Sons, Chichester (1978)
8. Hess, F.: Computing Riemann-Roch spaces in algebraic function fields and related topics. J. Symbolic Computation 33(4), 425–445 (2002)
9. The Magma computational algebra system, http://magma.maths.usyd.edu.au/
10. Milne, J.S.: Abelian varieties. In: Arithmetic geometry (Storrs, Conn., 1984), pp. 103–150. Springer, Heidelberg (1986)
11. Oort, F., Ueno, K.: Principally polarized abelian varieties of dimension two or three are Jacobian varieties. J. Fac. Sci. Univ. Tokyo Sect. IA Math. 20, 377–381 (1973)
12. Recillas, S.: Jacobians of curves with $g^1{}_4$'s are the Prym's of trigonal curves. Bol. Soc. Mat. Mexicana (2) 19(1), 9–13 (1974)
13. Serre, J.-P.: Galois Cohomology. Springer Monographs in Mathematics. Springer, Heidelberg (2002)
14. Serre, J.-P.: Algebraic Curves and Class Fields. In: GTM, vol. 117, Springer, Heidelberg (1988)
15. Vakil, R.: Twelve points on the projective line, branched covers, and rational elliptic fibrations. Math. Ann. 320(1), 33–54 (2001)
16. Vélu, J.: Isogénies entre courbes elliptiques. C. R. Acad. Sci. Paris, Séries A 273, 305–347 (1971)

On the Indifferentiability of the Sponge Construction

Guido Bertoni[1], Joan Daemen[1], Michaël Peeters[2], and Gilles Van Assche[1]

[1] STMicroelectronics
[2] NXP Semiconductors
http://sponge.noekeon.org/

Abstract. In this paper we prove that the sponge construction introduced in [4] is indifferentiable from a random oracle when being used with a random transformation or a random permutation and discuss its implications. To our knowledge, this is the first time indifferentiability has been shown for a construction calling a random permutation (instead of an ideal compression function or ideal block cipher) and for a construction generating outputs of any length (instead of a fixed length).

1 Introduction

All cryptographic hash functions of any significance known today, i.e., MD4, MD5, the SHA and RIPEMD [15] families and several others, share the same design paradigm. They all consist of the iterated application of a compression function. The iteration mechanism is known as Merkle-Damgård [8,16] and guarantees that if the compression function is collision-resistant, the resulting hash function is collision-resistant. This is a very attractive property as collision-resistance appears to be one of the most important properties of cryptographic hash functions. The compression functions of the above mentioned hash functions were designed with collision-resistance in mind. During the last years, with the recent collision attacks on SHA-1 as culminating point, it has become clear that designing a compression function that is both collision-resistant and efficient is not an easy task. Moreover, weaknesses have been shown in the Merkle-Damgård construction itself. While it does guarantee certain properties such as collision-resistance on the condition that the underlying compression function has the same property, this is not the case for all properties that are expected from cryptographic hash functions. A well known example of such a weakness, discussed in [7], is the insecurity of the MAC function constructed from a Merkle-Damgård hash function by feeding the latter with the secret key followed by the message.

In [7] Coron et al. propose a number of variants of the Merkle-Damgård construction that do not have this and other weaknesses. For each of these constructions they provide theorems stating that if the compression function is constructed using an ideal component, i.e., a finite input length (FIL) random oracle or an ideal block cipher (used as Davies-Meyer compression function), the hash function behaves as a random oracle [3] with output truncated to a fixed length. They present

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 181–197, 2008.

their theorems in the indifferentiability framework that was introduced by Maurer et al. in [14]. As a (truncated) random oracle has all desired properties that may be expected from a cryptographic hash function, this provides a direction for the design of hash functions that do not only provide resistance against collision search, but are as strong as a truncated random oracle with respect to many criteria. Instead of constructing an efficient compression function that is collision-resistant, one shall now design an efficient function that behaves as a FIL random oracle, or in other words, a random $n+m$ to n bit compression function, or an ideal block cipher. In the meanwhile, several other hash function constructions have been shown to be indifferentiable from a random oracle, see for example [2,6]. Note that indifferentiability is not the only approach to proving properties of hash function constructions: some authors analyze the properties of the compression function that can be preserved by the construction [1,2].

Recently, we introduced a new iterative hash function construction, called a sponge [4]. It builds upon a fixed-length transformation (i.e., with codomain equal to domain) or permutation f instead of a compression function and can generate output strings of infinite length. In [4] we proved that when f is a random transformation or permutation, the resulting function is only distinguishable from a random oracle with probability below $N(N+1)/2^{c+1}$, where N is the number of calls to f (and f^{-1}) and c is a security parameter related to the size of the domain of f. At first sight, one may consider the indistinguishability proof as an argument that it behaves as a random oracle with probability $1-N(N+1)/2^{c+1}$. However, this is restricted to adversaries that can only query the sponge function and not f (and f^{-1}). In a concrete hash function, f is publicly specified and this is of limited interest. We also included computations of the complexity of a number of so-called critical operations and discussed how this impacts the classical properties expected from hash functions such as collision-resistance and (2nd)-preimage resistance. However, this does not result in lower bounds for the security of these properties but rather upper bounds to the reachable security level. In this paper we apply the approach of [7] to the sponge construction and demonstrate that the advantage of an adversary in differentiating the sponge construction from a random oracle is about $N(N+1)/2^{c+1}$ if the underlying f is a random transformation and an even smaller upper bound if it is a random permutation.

As discussed above, this implies that the sponge construction when calling a random transformation or permutation has all properties of a random oracle as long as c is large enough. Hence we are now able to provide the security bounds for collision-resistance and (2nd)-preimage resistance that are lacking in [4].

There are several iterative hash function constructions that have been shown to be indifferentiable from a random oracle. However, the sponge construction has two unique features. First, it can generate long outputs. While other constructions can only behave as a random oracle whose output has been truncated to a fixed length, a random sponge does not have this limitation and may also serve as a reference for stream ciphers. Second, it can be built using a permutation, where both f and f^{-1} can be queried by the adversary. Paradoxically, collision-resistance and (2nd)-preimage resistance can be realized by employing a function that is easy to invert.

In [4] our main goal was to define a reference for security properties of hash designs. With our indifferentiability result, we prove that the resistance of the sponge construction calling a random transformation or permutation is as good as that of a random oracle, lower bounded by about $N(N+1)/2^{c+1}$. This coincides with what is presented in [4] as the [illegible]. Despite our original intention in [4], we argue that the sponge construction can lead to practical hash function designs. First of all, as mentioned in [4], the support for long outputs is a useful feature for a hash function when being used as a mask generating function (MGF) or a key derivation function (KDF). Second, instead of a collision-resistant compression function (Merkle-Damgård) or a random-looking compression function or ideal block cipher (as in [7]), it takes the design of a random-looking permutation. As a good block cipher should behave as a set of (independent and) random-looking permutations, hash function design can now benefit from insights gained in block cipher design. However, as opposed to a block cipher, a permutation has no key schedule and has not the concerns that come with it such as its computational overhead and possible related-key weaknesses. This makes in our opinion the sponge construction a very interesting alternative to the constructions based on a compression function.

The remainder of this paper is organized as follows. Section 2 gives a short introduction to indifferentiability applied to hash function constructions and is followed by Section 3 that defines and discusses the sponge construction in the indifferentiability setting. Section 4 gives the actual proofs and finally Section 5 discusses its implications.

2 Indifferentiability from a Random Oracle

Indifferentiability deals with the interaction between systems where the objective is to show that two systems cannot be told apart by an adversary able to query both systems but not knowing a priori which system is which. For hash function constructions, a random oracle serves as an ideal system.

We use the definition of random oracle from [3]. A random oracle, denoted $\mathcal{RO}$, takes as input binary strings of any length and returns for each input a random infinite string, i.e., it is a map from $\mathbf{Z}_2^*$ to $\mathbf{Z}_2^\infty$, chosen by selecting each bit of $\mathcal{RO}(x)$ uniformly and independently, for every x. In [7] and other papers on the subject, one does not consider indifferentiability from a random oracle, but rather a random oracle with output truncated to a fixed number of bits.

The indifferentiability framework was introduced by Maurer et al. in [14] and is an extension of the classical notion of indistinguishability. Coron et al. applied it to iterated hash function constructions in [7] and demonstrated for a number of iterated hash function constructions that they are indifferentiable from a random oracle if the compression function is a random FIL oracle. In this section we give a brief introduction to these subjects; for a more in-depth treatment, we refer to the original papers.

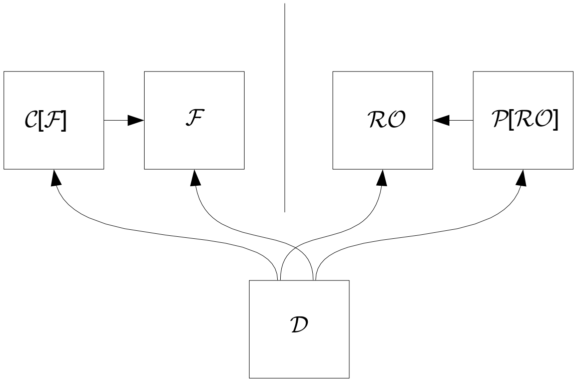


Fig. 1. The differentiability setup

In the context of iterated hashing the adversary shall distinguish between two systems that each have two components, as illustrated in Figure 1. The system at the left is the combination of the ideal compression function $\mathcal{F}$ and the hash function construction $\mathcal{C}$. The adversary can make queries to both components separately, where the latter in turn calls the former to construct its responses. This is denoted by $\mathcal{C}[\mathcal{F}]$. These are the two different interfaces to the system to the left.

The system at the right consists of a random oracle (with truncated output) $\mathcal{RO}$ providing the same interface as $\mathcal{C}[\mathcal{F}]$. To be indifferentiable from the system at the left, the system at the right also needs a subsystem offering the same interface to the adversary as the ideal compression function $\mathcal{F}$. This is called a [illegible] $\mathcal{P}$ and its role is to simulate the ideal compression function $\mathcal{F}$ so that no distinguisher can tell whether it is interacting with the system at the left or with the one at the right. The output of $\mathcal{P}$ should look [illegible] with what the distinguisher can obtain from the random oracle $\mathcal{RO}$ as if $\mathcal{P}$ was $\mathcal{F}$ and $\mathcal{RO}$ was $\mathcal{C}[\mathcal{F}]$. To achieve that, the simulator can query the random oracle, denoted by $\mathcal{P}[\mathcal{RO}]$. Note that the simulator does not see the distinguisher's queries to the random oracle.

Indifferentiability of $\mathcal{C}[\mathcal{F}]$ from a random oracle $\mathcal{RO}$ is now satisfied if there exists a simulator $\mathcal{P}$ such that no distinguisher can tell the two systems apart with non-negligible probability, based on their responses to queries it may send. We repeat here the definition as given in [7] where the hash function construction is called Turing machine $\mathcal{C}$, the ideal compression function is called ideal primitive $\mathcal{F}$ and the random oracle is called ideal primitive $\mathcal{G}$.

Definition 1 ([7]). [illegible] $\mathcal{C}$ [illegible] $\mathcal{F}$ [illegible] (t_D, t_S, q, ϵ) [illegible] $\mathcal{G}$ [illegible] $\mathcal{P}[\mathcal{G}]$ [illegible] $\mathcal{D}$ [illegible]

$$|\Pr[\mathcal{D}[\mathcal{C}[\mathcal{F}], \mathcal{F}] = 1] - \Pr[\mathcal{D}[\mathcal{G}, \mathcal{P}[\mathcal{G}]] = 1]| < \epsilon. \quad (1)$$

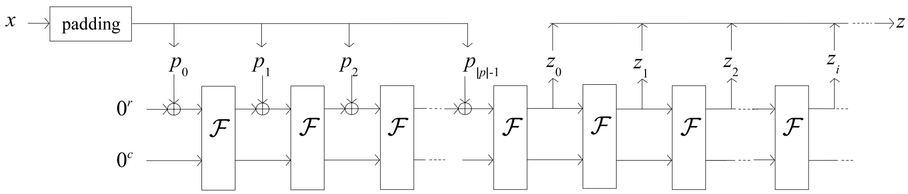


Fig. 2. The padded sponge construction

$\mathcal{G}$ t_S t_D q $\mathcal{C}[\mathcal{F}]$ $\mathcal{G}$, ϵ k

Now, it is shown in [14] that if $\mathcal{C}[\mathcal{F}]$ is indifferentiable from a random oracle, then $\mathcal{C}[\mathcal{F}]$ can replace the random oracle in any cryptosystem, and the resulting cryptosystem is at least as secure in the ideal compression function model as in the random oracle model. This is much stronger than the indistinguishability of $\mathcal{C}[\mathcal{F}]$ from a random oracle, which just merely means that an attacker that can query $\mathcal{C}[\mathcal{F}]$, but has no direct access to $\mathcal{F}$, cannot distinguish it from a random oracle. As said, for hash function constructions indistinguishability makes little sense as, for any concrete hash function, the compression function $\mathcal{F}$ is public and hence accessible to the adversary.

3 The Sponge Construction

3.1 Definition

In this section we define the sponge construction. Our definition is a special case of the more general definition in [4]. To simplify the presentation, we restrict the input and output of the sponge to binary strings instead of a more general alphabet. Our indifferentiability result can however easily be extended to the generic definition. The (padded) sponge construction is illustrated in Figure 2.

In the sequel, we generally denote by x a message in $\mathbf{Z}_2^*$, and by p a sequence of blocks of r bits each (i.e., $p \in \mathbf{Z}_2^{r*}$), indexed from 0 to $|p|-1$, with $|p|$ the number of r-bit blocks of p.

Definition 2. r, c sponge function $\mathcal{S}[\mathcal{F}]$ r $\mathbf{Z}_2^{r*}$ $\mathbf{Z}_2^{\infty}$ $\mathcal{F}$ $\mathbf{Z}_2^{r+c} = \mathbf{Z}_2^r \times \mathbf{Z}_2^c$ c capacity r bitrate[1] $\oplus$ r 0^r

[1] The bitrate r is not to be confused with rate meaning the number of block cipher calls it takes to implement the compression function, as in, e.g., [12].

p 0^r $|p| > 0$ $p_{|p|-1} \neq 0^r$

Algorithm 1. The sponge construction $\mathcal{S}[\mathcal{F}]$

Input $p = p_0 p_1 \ldots p_{|p|-1}$ and requested length n
Require: $|p| \geq 1$ and $p_{|p|-1} \neq 0^r$
Output $z \in \mathbf{Z}_2^n$
$s = (s_\mathrm{a}, s_\mathrm{c}) = (0^r, 0^c)$
for $i = 0$ to $|p| - 1$ **do**
  $(s_\mathrm{a}, s_\mathrm{c}) = \mathcal{F}(s_\mathrm{a} \oplus p_i, s_\mathrm{c})$
end for
for $i = 0$ to $\lceil \frac{n}{r} \rceil - 1$ **do**
  Append s_a to the output
  $(s_\mathrm{a}, s_\mathrm{c}) = \mathcal{F}(s_\mathrm{a}, s_\mathrm{c})$
end for
Discard the last $r\lceil \frac{n}{r} \rceil - n$ bits

3.2 Graph Representation of Sponge Operation

In [4] we used a graph representation to prove bounds on success probability of generating collisions. We adopt this graph representation in the specification of our simulators. In our discussions on the graphs, we need to clearly distinguish between the first r bits and the last c bits of an $r + c$-bit variable s. For this, we again use the notation of [4]: $A = \mathbf{Z}_2^r$ and $C = \mathbf{Z}_2^c$ and we call the first r bits of s its A-part s_a, and the last c bits its C-part s_c.

We consider the transformation $\mathcal{F}$ as a directed graph whose vertex set (called) is $A \times C$ and whose edges are $(s, \mathcal{F}(s))$. It has both 2^{r+c} nodes and edges. From the node graph, we derive the (directed) supernode graph, with vertex set (called) equal to C. In this graph, an edge $(s_\mathrm{c}, t_\mathrm{c})$ is in the edge set if and only if $\exists s_\mathrm{a}, t_\mathrm{a}$ such that $((s_\mathrm{a}, s_\mathrm{c}), (t_\mathrm{a}, t_\mathrm{c}))$ is an edge in the node graph. The set of supernodes is a partition of the nodes where a supernode contains the 2^r nodes with the same C-part.

The sponge construction operates on a chaining variable s and its operation can be seen as a walk through the node graph of the chaining variable. We denote the chaining variable before processing p_i by s_i. Its initial value is $s_0 = (0^r, 0^c)$. Then for each block p_i, it performs a two-step transition. First, it moves to the node s' within the same supernode with $s'_\mathrm{a} = s_{i,\mathrm{a}} \oplus p_i$, and then it follows the edge starting from s', arriving in s_{i+1}. After processing all blocks of p it is in node $s_{|p|}$. Then it gives out the A-part of $s_{|p|}$ as z_0. For each additional block z_i produced, it follows the edge from $s_{|p|+i-1}$ arriving in $s_{|p|+i}$ and gives out the A-part of the latter as z_i. Note that this can be considered a special case of the above two-step transition if we extend p with blocks $p_{|p|+i} = 0^r$ for all $i \geq 0$. Clearly, the chaining variable s_i is completely determined by the first i blocks of p. We call this a to s_i. Or more exactly:

Definition 3 ([4]). [illegible] $(0^r, 0^c)$ [illegible] p [illegible] $s = (s_a, s_c)$ [illegible] $((s_a \oplus a, s_c), t)$ [illegible] $p' = pa$ [illegible] t

Note that although a path completely determines a node, there may be many paths to a node.

It follows from the above that z_j of $z = \mathcal{S}[\mathcal{F}](p)$ is the A-part of the node with path $p0^{rj}$. And so, given a path p (different from 0^{rj}) to a node s, one can find its A-part by a call to the sponge construction. We have $s_\mathrm{a} = z_j$ with $z = \mathcal{S}[\mathcal{F}](p')$ and p' and j given by $p = p'0^{rj}$ such that p' is a valid sponge input, i.e., $|p'| > 0$ and $p'_{|p'|-1} \neq 0^r$. For a path of form 0^{rj} there is no such p' and hence the sponge construction cannot be queried to obtain s_a.

3.3 The Padded Sponge Construction

The sponge construction $\mathcal{S}[\mathcal{F}](p)$ only supports input strings $p \in \mathbf{Z}_2^{r*}$ where p is not the empty string and has last block different from 0^r. To allow the input to be any binary string in $\mathbf{Z}_2^*$, one needs to define an injective mapping $\mathrm{pad}(x)$ that converts any binary string x to a valid sponge input. The simplest such mapping $\mathrm{pad}(x)$ consists in padding the string with a single bit 1 and a number w of zeroes with $0 \leq w < r$ so that $\mathrm{pad}(x)$ contains a multiple of r bits. To indicate the sponge construction including the padding operation, we use the symbol $\mathcal{S}'$:

$$\mathcal{S}'[\mathcal{F}](x) \triangleq \mathcal{S}[\mathcal{F}](\mathrm{pad}(x)).$$

3.4 The Distinguisher's Setting

We give proofs of indifferentiability for the cases that $\mathcal{F}$ is a random transformation or a random permutation. A random transformation (permutation) operating on a certain domain is a transformation selected randomly and uniformly from all transformations (permutations) operating on that domain.

The adversary shall distinguish between two systems using their responses to sequences of queries. At the left is the system $(\mathcal{S}'[\mathcal{F}], \mathcal{F})$. The padded sponge construction $\mathcal{S}'[\mathcal{F}]$ provides one interface denoted by $\mathcal{H}$, taking a binary string $x \in \mathbf{Z}_2^*$ and an integer n and returning a binary string $y \in \mathbf{Z}_2^n$, the sponge output truncated to n bits. If $\mathcal{F}$ is a random transformation it has a single interface $\mathcal{F}^1$ which takes as input an element s of $\mathbf{Z}_2^{r+c}$ and returns $t = \mathcal{F}(s)$, an element of the same set. If $\mathcal{F}$ is a random permutation, it has an additional interface $\mathcal{F}^{-1}$ that implements the inverse of $\mathcal{F}$. Note that the sponge construction in Algorithm 1 only uses the interface $\mathcal{F}^1$.

At the right is the system $(\mathcal{RO}, \mathcal{P}[\mathcal{RO}])$. It offers the same interface as the left system, i.e., $\mathcal{RO}$ provides the interface $\mathcal{H}$ and returns an output truncated to the requested length. We define two simulators, one for the case of a random transformation and another one for the case of a random permutation. The transformation simulator provides a single interface $\mathcal{F}^1$. The permutation simulator provides both interfaces $\mathcal{F}^1$ and $\mathcal{F}^{-1}$.

Let $\mathcal{X}$ be either $(\mathcal{S}'[\mathcal{F}], \mathcal{F})$ or $(\mathcal{RO}, \mathcal{P}[\mathcal{RO}])$. The sequence of queries Q to $\mathcal{X}$ consist of a sequence of queries to the interface $\mathcal{H}$, denoted Q^0 and a sequence of queries to the interface $\mathcal{F}^1$ (and $\mathcal{F}^{-1}$), denoted Q^1. Q^0 is a sequence of couples (x, n), with $x \in \mathbf{Z}_2^*$ and n a positive integer. Q^1 is a sequence of couples (s, b) with $s \in \mathbf{Z}_2^{r+c}$ and b either 1 or -1, indicating whether the interface $\mathcal{F}^1$ or $\mathcal{F}^{-1}$ is addressed. In the case that $\mathcal{F}$ is a transformation, b is restricted to 1.

3.5 The Cost of Queries

Definition 1 suggests expressing an upper bound to the advantage of a distinguisher in terms of the number of queries q. The bounds provided in [7] however also make use of parameter ℓ, the maximum input length of the queries. In our bounds we use another measure for the query complexity which is more natural when applied to the sponge construction. We call this measure [illegible] and denote it by N. The cost N of queries to a system $\mathcal{X}$ is the total number of calls to $\mathcal{F}$ or $\mathcal{F}^{-1}$ it would yield if $\mathcal{X} = (\mathcal{F}, \mathcal{S}'[\mathcal{F}])$, either directly due to queries Q^1, or indirectly via queries Q^0 to $\mathcal{S}'[\mathcal{F}]$. The cost of a sequence of queries is fully determined by their number and their input and output lengths. Each query to $\mathcal{F}^1$ or $\mathcal{F}^{-1}$ contributes 1 to the cost. A query to $\mathcal{H}$ with an ℓ-bit input contributes $\lfloor \frac{\ell}{r} \rfloor + \lceil \frac{n}{r} \rceil$ to the cost (assuming the simple padding of Section 3.3 is used). Our bounds in terms of cost are comparable to those of [7]: for a fixed output size, as considered in [7], N is an affine function of q and $q\ell$.

In the sequel, we consider the indifferentiability as in Definition 1 but with the cost N replacing the number of queries q and their maximum length ℓ.

4 Indifferentiability Proofs

4.1 The Simulators We Use in Our Proofs

We define simulators for the case that $\mathcal{F}$ is a random transformation and for the case of a random permutation. In both cases, the simulator should behave as a deterministic function and give responses to queries Q^1 that in combination with the responses to queries Q^0 to the random oracle shall minimize the probability that the system $(\mathcal{RO}, \mathcal{P}[RO])$ can be distinguished from a system $(\mathcal{S}'[\mathcal{F}], \mathcal{F})$. In this section we informally explain how our simulators work.

A simulator keeps track of the queries it received and the responses it returned in a graph, very similar to the graphs discussed in Section 3.2. The only difference is that initially the simulator graph has no edges and for each new query $\mathcal{F}^1(s)$ (or $\mathcal{F}^{-1}(s)$) the simulator generates a response t and adds the edge (s, t) (or (t, s)). Note that using the responses of the simulator to its queries, the adversary can fully reconstruct the simulator graph.

In order to motivate the design of the simulators, we now discuss properties of this graph that it has at any moment during or after the queries, using an example depicted in Figure 3.

For a subset of the nodes in the simulator graph, the adversary knows a path. From Definition 3, it is clear that these are the nodes that have an incoming edge

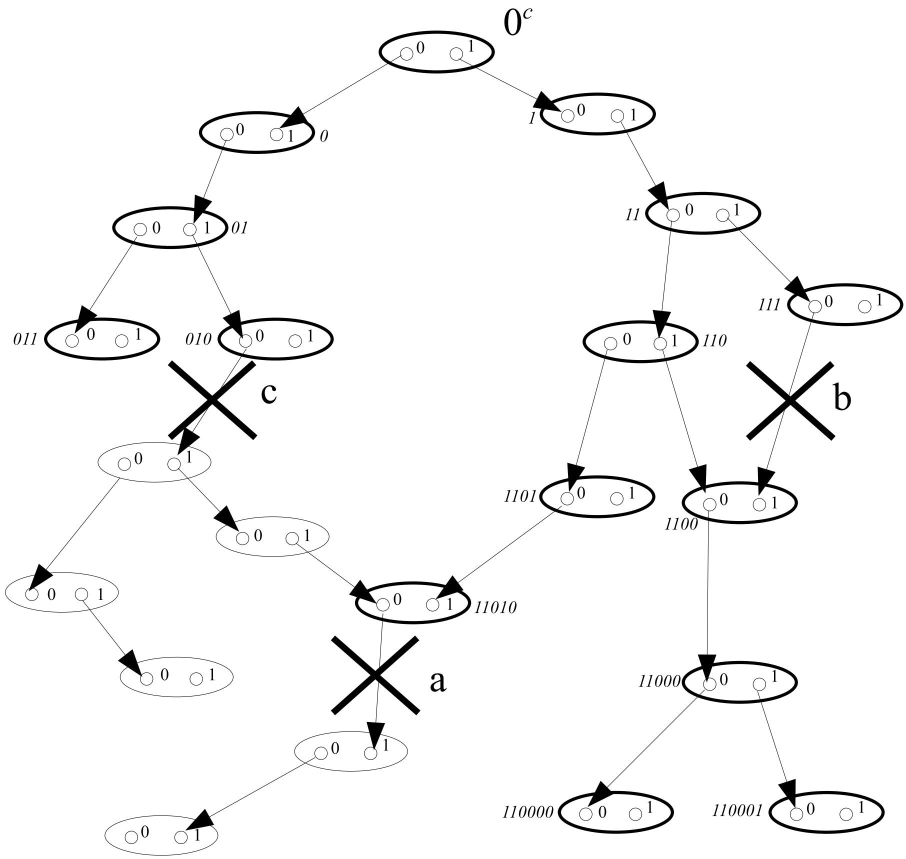


Fig. 3. Example of simulator graph. The rooted supernodes are in bold. Paths are indicated in italic next to the nodes having a path.

and are in a supernode that can be reached from supernode 0^c by following the directed edges from supernode to supernode. For this purpose, we define the set of [illegible] supernodes R as the subset of C containing 0^c and all the supernodes accessible from it through the supernode graph. By extension, we say that a node $s = (s_{\mathrm{a}}, s_{\mathrm{c}})$ is rooted if $s_{\mathrm{c}} \in R$. So the adversary knows paths to all rooted nodes that have an incoming edge from another rooted node, plus the empty path of the $(0^r, 0^c)$ node. For each of these rooted nodes it can query the interface $\mathcal{H}$ of the system hoping to reveal an inconsistency, which is evidence that it is not $(\mathcal{S}'[\mathcal{F}], \mathcal{F})$. We call [illegible] the responses to a sequence of queries Q that do not result in such inconsistency.

We will now explain why our simulators generate sponge-consistent responses (up to 2^c queries Q^1). Whenever a simulator receives a query $\mathcal{F}^1(s)$ with s rooted, it will result in an image t with known path. Therefore, the simulator constructs the A-part of t to be sponge-consistent by querying $\mathcal{RO}$ using the path to t

(except for the all-zero path). When the simulator receives a query $\mathcal{F}^1(s)$ with s not rooted, no path to the image t is known and it chooses t randomly from all the nodes (with no incoming edge, if $\mathcal{F}$ is a random permutation).

The idea is that the simulators are designed so that a call to $\mathcal{F}^1(s)$ results only in the path of a single node becoming known, that of $t = \mathcal{F}(s)$ if s is rooted. For that, when selecting t_c for a rooted node s, they exclude the supernodes with outgoing edges (cases a and c in Figure 3). Additionally, they avoid the occurrence of nodes with multiple paths. For that, when selecting t_c for a rooted node s, they exclude the rooted supernodes (case b in Figure 3) and those with outgoing edges (case c in Figure 3). The permutation simulator avoids paths of nodes becoming known as a result of a call to $\mathcal{F}^{-1}(s)$ altogether by excluding rooted supernodes when selecting t_c.

Let O be the set of supernodes with an outgoing edge. When the simulator receives a query $\mathcal{F}^1(s)$ with s a rooted node and all supernodes are rooted or have an outgoing edge, i.e., if $R \cup O = C$, it can no longer ensure sponge-consistency and we call the simulator [illegible]. As every query to the simulator adds at most one edge and that hence $R \cup O$ can be extended by at most 1 per query, this cannot happen before 2^c queries.

4.2 When Being Used with a Random Transformation

The simulator for the case that $\mathcal{F}$ is a random transformation is given in Algorithm 2. We prove the indifferentiability by means of a series of lemmas and a final theorem.

Algorithm 2. The transformation simulator $\mathcal{P}[\mathcal{RO}]$

Interface $\mathcal{F}^1$, taking node s as input
if node s has no outgoing edge **then**
    if node s is rooted AND $R \cup O \neq C$ (no saturation) **then**
        Construct path to t: find path to s, append s_a and call the result p
        Write p as $p = p'0^{rj}$ where p' does not end with 0^r
        if p' can be unpadded into x **then**
            Assign to t_a the value z_j with $z = \mathcal{RO}(x)$
        else
            Choose t_a randomly and uniformly
        end if
        Choose t_c randomly and uniformly from $C \setminus (R \cup O)$
        Let $t = (t_a, t_c)$
    else
        Choose t randomly and uniformly from all nodes
    end if
    Add an edge from s to t
end if
return the node t at the end of the outgoing edge from s

Lemma 1. [illegible]

[illegible] First, we show that the rooted supernodes in the supernode graph form a tree. When no edges exist, this is indeed the case. The only way to create a new rooted node is by calling $\mathcal{F}^1(s)$ with s rooted. Assuming the simulator is not saturated, this happens only in first part of Algorithm 2 (lines 4–12), if s is rooted and has no outgoing edge. The new edge only adds a single supernode to R as the simulator selects it from the supernodes with no outgoing edges. Moreover, the new edge cannot arrive in a rooted supernode (because the simulator selects t_c from $C \setminus R$) or in a supernode from which a rooted supernode can be reached (because the simulator select t_c from the supernodes with no outgoing edges).

Then, for two connected supernodes (s_c, t_c), there exists only one edge in the simulator graph of the form $((s_a, s_c), (t_a, t_c))$. This is because the simulator chooses a distinct C-part for each new rooted node (unless it is saturated).

Finally, since A is a group, each r-bit block of the path is uniquely determined by the transitions on the A-part of the nodes. □

For a given set of queries Q and their responses $\mathcal{X}(Q)$, we define the [illegible] as the property that the responses to Q^0 are equal to those that one would obtain by applying the sponge construction from the responses to Q^1 (when the queries Q^1 suffice to perform this calculation), i.e., that $\mathcal{X}(Q^0) = \mathcal{S}'[\mathcal{X}(Q^1)](Q^0)$. By construction, the queries, and their responses, made to the system $(\mathcal{S}'[\mathcal{F}], \mathcal{F})$ are sponge-consistent. For the sponge-consistency of the queries, and their responses, made to $(\mathcal{RO}, \mathcal{P}[\mathcal{RO}])$, we refer to the following lemma.

Lemma 2. [illegible] $\mathcal{P}[\mathcal{RO}]$ [illegible] $\mathcal{RO}$ [illegible]

[illegible] The adversary can check by querying the random oracle for sponge-consistency for every node s in the simulator graph to which it knows the path p. The all-zero path does not correspond to a block that can be output by the sponge construction, so without loss of generality we assume that $p \neq 0^{rj}$.

Given the path p to the node s, its A-part must be equal to z_j with $z = \mathcal{RO}(x)$, where $\text{pad}(x) = p'$ and p' is a valid sponge input given by $p = p'0^{rj}$. As Lemma 1 says, there is only a single path to any rooted node in the simulator graph, and thus the simulator guarantees this equality for the response t to every query to $\mathcal{F}^1(s)$ with s a rooted node, as long as it is not saturated.

We also need to show that no path is assigned to a node unless its A-part is chosen by the lines 6–9 of Algorithm 2. Indeed, the supernode t_c (at line 11) is the only supernode that becomes rooted due to the query. This is because the simulator excludes supernodes with outgoing edges in the selection of t_c (as long as the simulator is not saturated).

It follows that the simulator guarantees sponge-consistency for all queries Q up to saturation. □

Lemma 3. [illegible] Q^0 [illegible] 2^c [illegible] Q^1 [illegible] Q^1 [illegible] Q^0

[illegible] A query in Q^0 consists of an input x and a length n. Let $p = \text{pad}(x)0^{r\lceil \frac{n}{r} \rceil}$ and we can now convert this query into $|p|$ queries to $\mathcal{F}^1$. Let $s_0 = (0^r, 0^c)$ and $s_{i+1} = \mathcal{F}^1(s_{i,\text{a}} \oplus p_i, s_{i,\text{a}})$ for $0 \leq i < |p|$ be the responses to the new queries. As Lemma 2 says that all queries up to cost 2^c are sponge-consistent, the output to the original (x, n) query consists of the concatenation of the A-parts of $s_{|p|}$ to $s_{|p|+\lceil \frac{n}{r} \rceil - 1}$ truncated to n bits. By the definition of the cost of queries, the original query in Q^0 has cost $|p|$ and it results in $|p|$ queries in Q^1, each one with cost 1.

This process can be repeated for all queries in Q^0 resulting in a sequence of queries Q^1 with the same cost. If there are queries in Q^0 with inputs having common prefixes, these can give rise to the same queries in Q^1 resulting in a reduction in cost. □

Lemma 4. [illegible] $\mathcal{F}$ [illegible] $\mathcal{P}[\mathcal{RO}]$ [illegible] $N < 2^c$ [illegible] Q^1 [illegible]

$$f_{\text{T}}(N) = 1 - \prod_{i=1}^{N} \left(1 - \frac{i}{2^c}\right) .$$

[illegible] The advantage is defined as

$$\text{Adv}(\mathcal{A}) = |\Pr[\mathcal{A}[\mathcal{F}] = 1] - \Pr[\mathcal{A}[\mathcal{P}[\mathcal{RO}]] = 1]|$$

The response sequence x to a sequence of N different queries is a sequence of N values in $A \times C$. We can provide an upper bound of the advantage by computing the probability distributions of the outcomes of the queries to $\mathcal{F}$ on the one hand and to $\mathcal{P}[\mathcal{RO}]$ on the other. The optimal adversary gives back 1 for the response sequence x if $\Pr(x|\mathcal{F}) > \Pr(x|\mathcal{P}[\mathcal{RO}])$ and 0 otherwise, yielding the following upper bound:

$$\text{Adv}(\mathcal{A}) \leq \frac{1}{2} \sum_x |\Pr(x|\mathcal{F}) - \Pr(x|\mathcal{P}[\mathcal{RO}])|, \tag{2}$$

where the righthand side of this equation is known as the variational distance. Since $\mathcal{F}$ is a transformation over $A \times C$ chosen randomly and uniformly, the responses to the different queries are independent and uniformally distributed over $A \times C$. It follows that all $(2^r 2^c)^N$ possible outcomes are all equiprobable.

By inspecting Algorithm 2, the simulator always returns uniform values for the A-part of the image. For the C-part, the simulator chooses it non-uniformly only if the pre-image s is rooted. To obtain the greatest possible variational distance, the optimum strategy consists in creating N rooted nodes. As a response to the first query, it may return all values but 0^r. At each subsequent query, one value of C is added to R, and thus for each query, the simulator returns a C-part value different from 0^r and all previous ones. Note that by restricting $N < 2^c$

the simulator will not be saturated. Using this strategy gives us an upper bound on the variational distance. So for the simulator, there are $(2^r)^N(2^c-1)_{(N)}$ (where $a_{(n)}$ denotes $a!/(a-n)!$) possible responses with different C-parts, each with equal probability $((2^r)^N(2^c-1)_{(N)})^{-1}$, and the $(2^r)^N((2^c)^N-(2^c-1)_{(N)})$ others have probability 0. This gives:

$$\mathrm{Adv}(\mathcal{A}) \leq 1 - \frac{(2^c-1)_{(N)}}{(2^c)^N} = 1 - \prod_{i=1}^{N}\left(1-\frac{i}{2^c}\right) . \tag{3}$$

□

We have now all ingredients to prove the following theorem.

Theorem 1. [illegible] $\mathcal{S}'[\mathcal{F}]$ [illegible] (t_D, t_S, N, ϵ) [illegible] t_D $t_S = O(N^2)$ $N < 2^c$ [illegible] ϵ [illegible] $\epsilon > f_{\mathrm{T}}(N)$

[illegible] As discussed in Lemma 3 we can construct from a set of query sequences Q^0, Q^1 an equivalent sequence of queries $Q^{1\prime} \circ Q^1$ with no higher cost and giving at least the same information. So, without loss of generality, we only need to consider adversaries using queries $\overline{Q}^1 = Q^{1\prime} \circ Q^1$ and their response $\mathcal{X}(\overline{Q}^1)$ and no queries Q^0.

For any fixed query $\overline{Q}^1$, we look at the problem of distinguishing the random variable $\mathcal{F}(\overline{Q}^1)$ from the random variable $\mathcal{P}[\mathcal{RO}](\overline{Q}^1)$. For a sequence of queries $\overline{Q}^1$ with cost N, Lemma 4 upper bounds the advantage of such an adversary to $f_{\mathrm{T}}(N)$.

We have $t_S = O(N^2)$ as for each query to the simulator with s rooted, it must find the path to s and send a query to the random oracle of cost equal to the length of the path to s. The length of the path to s it upper bounded by N, the total number of rooted supernodes in the simulator graph. □

If N is significantly smaller than 2^c, we can use the approximation $1 - x \approx e^{-x}$ for $x \ll 1$ to simplify the expression for $f_{\mathrm{T}}(N)$:

$$f_{\mathrm{T}}(N) \approx 1 - e^{-\frac{N(N+1)}{2^{c+1}}} < \frac{N(N+1)}{2^{c+1}}. \tag{4}$$

4.3 When Being Used with a Random Permutation

The simulator for the case that $\mathcal{F}$ is a random permutation is given in Algorithm 3. We now can prove indifferentiability using a series of similar lemmas.

The proofs of Lemma 1 and Lemma 2 are valid for the permutation simulator with respect to all calls to $\mathcal{F}^1$ but do naturally not consider calls to $\mathcal{F}^{-1}$. The proofs can simply be extended to the permutation simulator case by noting that the $\mathcal{F}^{-1}$ interface of the simulator excludes rooted nodes in the selection of the response, implying that a call to $\mathcal{F}^{-1}$ cannot lead to new rooted nodes and hence also not to new paths. The proof of Lemma 3 is valid for the permutation

Algorithm 3. The permutation simulator $\mathcal{P}[\mathcal{RO}]$

Interface $\mathcal{F}^1$, taking node s as input
if node s has no outgoing edge **then**
  if node s is rooted AND $R \cup O \neq C$ (no saturation) **then**
    Construct path to t: find path to s, append s_a and call the result p
    Write p as $p = p'0^{rj}$ where p' does not end with 0^r
    if p' can be unpadded into x **then**
      Assign to t_a the value z_j with $z = \mathcal{RO}(x)$
    else
      Choose t_a randomly and uniformly
    end if
    Choose t_c randomly and uniformly from $C \setminus (R \cup O)$ and such that $(t_\mathrm{a}, t_\mathrm{c})$ has no incoming edge yet
    Let $t = (t_\mathrm{a}, t_\mathrm{c})$
  else
    Choose t randomly and uniformly from all nodes that have no incoming edge yet
  end if
  Add an edge from s to t
end if

Interface $\mathcal{F}^{-1}$, taking node s as input
if node s has no incoming edge **then**
  Choose t_a randomly and uniformly
  Choose t_c randomly and uniformly from $C \setminus R$ and such that $(t_\mathrm{a}, t_\mathrm{c})$ has no outgoing edge yet
  Let $t = (t_\mathrm{a}, t_\mathrm{c})$
  Add an edge from t to s
end if

---

simulator as it is. Finally, the output produced by the interfaces $\mathcal{F}^1$ and $\mathcal{F}^{-1}$ are consistent, i.e., if $\mathcal{F}^1(s) = t$ then $\mathcal{F}^{-1}(t) = s$ and vice-versa.

Instead of Lemma 4 we now have the following lemma.

Lemma 5. [illegible] $\mathcal{F}$ [illegible] $\mathcal{P}[\mathcal{RO}]$ [illegible] $N < 2^c$ [illegible] Q^1 [illegible]

$$f_\mathrm{P}(N) = 1 - \prod_{i=0}^{N-1} \left(\frac{1 - \frac{i+1}{2^c}}{1 - \frac{i}{2^r 2^c}} \right) .$$

[illegible] The proof is similar to that of Lemma 4. Since $\mathcal{F}$ is a permutation over $A \times C$ chosen randomly and uniformly, the only limitation is that for the i-th query, the image (or preimage) shall not be equal to any of the found images (or preimage), resulting in $(2^r 2^c) - i$ possibilities. This leads to $(2^r 2^c)_{(N)}$ possible outcomes each with probability $((2^r 2^c)_{(N)})^{-1}$ and $(2^r 2^c)^N - (2^r 2^c)_{(N)}$ outcomes with probability 0.

From inspecting Algorithm 3 if follows that the adversary obtains the greatest possible variational distance when he creates N rooted nodes. This leads to the same distribution as for the transformation simulator. The possible outcomes of the permutation simulator are a subset of the possible outcomes for $\mathcal{F}$. This gives:

$$\mathrm{Adv}(\mathcal{A}) \leq 1 - \frac{(2^r)^N (2^c - 1)_{(N)}}{(2^r 2^c)_{(N)}} = 1 - \prod_{i=0}^{N-1} \left(\frac{1 - \frac{i+1}{2^c}}{1 - \frac{i}{2^r 2^c}} \right) . \tag{5}$$

□

These lemmas and proofs result in the following theorem, where the proof is similar to that of Theorem 1.

Theorem 2. [illegible] $\mathcal{S}'[\mathcal{F}]$ [illegible] (t_D, t_S, N, ϵ) [illegible] t_D $t_S = O(N^2)$ $N < 2^c$ [illegible] ϵ [illegible] $\epsilon > f_{\mathrm{P}}(N)$

If N is significantly smaller than 2^c, $f_P(N)$ can be approximated closely by:

$$f_P(N) \approx 1 - e^{-\frac{(1-2^{-r})N^2 + (1+2^{-r})N}{2^{c+1}}} < \frac{(1-2^{-r})N^2 + (1+2^{-r})N}{2^{c+1}} . \tag{6}$$

Note that using a random permutation results in a better bound than using a random transformation. By assigning distinct C-part values of rooted nodes, the simulators tend to generate an output distribution which is closer to that of a permutation than to that of a transformation.

5 Discussion and Conclusions

We have proven that the sponge construction calling a random transformation or permutation is indifferentiable from a random oracle and obtained concrete bounds. Here, the security parameter is the capacity c and not the output length of the hash function. Note that other constructions also consider the size of the internal state as a security parameter, e.g., [13].

One may ask the question: what does this say about resistance to classical attacks such as collision-resistance, including multicollisions [9], (2nd) preimage resistance, including long-message attacks [10] and herding [11]? In general, it is expected that a hash function offers the same resistance as would a random oracle, truncated to the hash function's output length n. The success probability after q queries is about $q^2/2^{n+1}$ for generating collisions and $q/2^n$ for generating a (2nd) preimage. The sponge construction does not have a fixed output length. However, when a hash function with the sponge construction is used in an actual cryptographic scheme, its output will be truncated. Our indifferentiability bounds in terms of the capacity c permit to express up to which output length n such a hash function may offer the expected resistance. For example, it offers collision resistance (as a truncated random oracle would) for any output length smaller than the capacity and (2nd) preimage resistance for any output length smaller than half the capacity. In other words, when for instance $c = 512$, a random sponge offers the same resistance as a random oracle but with a maximum of 2^{256} in complexity.

A function with the sponge construction can be used to build a MAC function (by just pre-pending the key to the input) or, thanks to its long output, to build a synchronous stream cipher (by taking as input the concatenation of a key and an IV). Alternatively, the sponge construction can be used as a reference for expressing security claims when building new such designs.

Note that the bounds we have provided only hold when the sponge construction makes use of a random transformation or random permutation. When a concrete transformation or permutation is taken, no such bounds can be given. See for example [5] and also [14] for discussions on this subject. However, our bounds do say that using the sponge construction excludes generic attacks with a success probability higher than the maximum of our bound $\frac{N(N+1)}{2^{c+1}}$ and the success probability the attack would have for a random oracle. By generic attacks we mean here attacks such as those described in [9,10,11], that do not exploit specific properties of the transformation or permutation used but only properties of the construction.

References

1. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
2. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM (ed.) ACM Conference on Computer and Communications Security 1993, pp. 62–73 (1993)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: Ecrypt Hash Workshop 2007 (May 2007), http://www.csrc.nist.gov/pki/HashWorkshop/Public_Comments/2007_May.html
5. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, pp. 209–218. ACM Press, New York (1998)
6. Chang, D., Lee, S., Nandi, M., Yung, M.: Indifferentiable security analysis of popular hash function with prefix-free padding. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 283–298. Springer, Heidelberg (2006)
7. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
8. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
9. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
10. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)

11. Kohno, T., Kelsey, J.: Herding hash functions and the Nostradamus attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 222–232. Springer, Heidelberg (2006)
12. Kuwakado, H., Morii, M.: Indifferentiability of single-block-length and rate-1 compression functions, Cryptology ePrint Archive, Report 2006/485 (2006), http://eprint.iacr.org/
13. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
14. Maurer, U., Renner, R., Holenstein, C.: Inidifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
15. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
16. Merkle, R.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)

A New Mode of Operation for Block Ciphers and Length-Preserving MACs

Yevgeniy Dodis[1], Krzysztof Pietrzak[2], and Prashant Puniya[1]

[1] New-York University, USA
{dodis,puniya}@cs.nyu.edu
[2] CWI Amsterdam, The Netherlands
pietrzak@cwi.nl

Abstract. We propose a new mode of operation, *enciphered CBC*, for domain extension of length-preserving functions (like block ciphers), which is a variation on the popular CBC mode of operation. Our new mode is twice slower than CBC, but has many (property-preserving) properties not enjoyed by CBC and other known modes. Most notably, it yields the first constant-rate Variable Input Length (VIL) MAC from any length preserving Fixed Input Length (FIL) MAC. This answers the question of Dodis and Puniya from Eurocrypt 2007. Further, our mode is a secure domain extender for PRFs (with basically the same security as encrypted CBC). This provides a hedge against the security of the block cipher: if the block cipher is pseudorandom, one gets a VIL-PRF, while if it is "only" unpredictable, one "at least" gets a VIL-MAC. Additionally, our mode yields a VIL random oracle (and, hence, a collision-resistant hash function) when instantiated with length-preserving random functions, or even random permutations (which can be queried from both sides). This means that one does not have to re-key the block cipher during the computation, which was critically used in most previous constructions (analyzed in the ideal cipher model).

1 Introduction

Modes of operation allow one to build a Variable Input Length (VIL) primitive from a given Fixed Input Length (FIL) primitive. Currently, variants of two popular modes of operation are used to implements almost all known VIL primitives: the CBC mode, which operates on length preserving functions (like a block cipher), and the Merkle-Damgård (MD, aka as "cascade") mode, which operates on a compression function. In practice, the latter compression function h is often implemented out of a block cipher E via the Davies-Meyers transform: $h(x, y) = E_x(y) \oplus y$. Thus, one way or another, many useful primitives are built from a block cipher in practice. Unfortunately, we argue that neither the CBC nor the MD mode are entirely satisfactory and a new block cipher mode of operation is needed.

CBC MODE. Cipher Block Chaining (CBC) is a popular mode of operation for domain extension of pseudorandom functions (PRFs) [3], thus allowing one to build a MAC (recall that a PRF is a MAC) on roughly $n\ell$ bits by making ℓ calls to an n-bit block cipher E. However, here one must assume that E is a PRF, even if finally one is only interested in getting a MAC. Pseudorandomness is a much stronger assumption

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 198–219, 2008.

than unpredictability (which is all we need from a MAC). Thus, it is natural to ask if the CBC-MAC is secure if the block cipher is "only" *unpredictable*, in other words, if CBC is a good domain extender for MACs. Aside from being of great theoretical importance, a positive answer to this question would provide a hedge against the security of the block cipher E: if E is pseudorandom, one gets a VIL-PRF, while if it is only unpredictable, one at least gets a VIL-MAC. Unfortunately, An and Bellare [1] showed that this is not the case.[1] This motivates the following central question of this work:

Question 1. Is there a simple and efficient way to build a VIL-MAC from a length-preserving MAC (like an unpredictable block cipher)?

This question was recently explicitly addressed by Dodis and Puniya [14]. They argued that none of the existing techniques (as opposed to just CBC) give a satisfactory answer to this question (see [14] for a list of many failed approaches). They also presented the best-known-to-date solution. The idea is to use the Feistel network for $\omega(\log \lambda)$ rounds (where λ is the security parameter) to get a MAC from $2n$ to $2n$ bits. Then one can safely chop half of the output bits, getting a $2n \mapsto n$ bit MAC, after which one can apply any of the known efficient techniques to extend the domain of a "shrinking MAC" [1,16]. While elegant, this solution evaluates the given FIL-MAC $\omega(\ell \log \lambda)$ times to extend the domain of the FIL-MAC by a factor of ℓ. In contrast, the solution we present shortly will only use 2ℓ calls.

Coming back to CBC, another drawback of this mode is that it does not appear to be useful for building collision-resistant hash function (CRHFs) or random oracles (ROs) from block ciphers, even if the block cipher is modeled as an ideal cipher. Indeed, if the key to the block cipher is fixed and public, it is trivial to find collisions in the CBC mode, irrespective of the actual cipher.

MD MODE. Unlike the CBC mode, the MD mode seems to be quite universal, and variants of it were successfully used to argue domain extension results for many properties, including collision-resistance [11,18,21,8], pseudorandomness [5,6], unforgeability [1,16], indifferentiability from a random oracle [10], randomness extraction [12] and even "multi-property preservation" [7]. However, when using a block cipher, we will first have to construct a compression from the block cipher before we can apply MD.

One trivial way of doing this would to simply chop part of the output of E. However, this is very unsatisfactory on multiple levels. First, to achieve constant efficiency rate for the cascade construction, one must chop a constant fraction of the output bits. However, already chopping a super-logarithmic number of bits will not, in general, preserve the security of E as a MAC, making it useless for answering Question 1. Second, even for the case of PRFs and ROs, where chopping a linear fraction of bits does preserve the corresponding property, one loses a lot in exact security, since the output is now much shorter. For example, dropping half of the bits would give a VIL-PRF with efficiency rate 2 and security $\mu^2/2^{n/2}$ (where μ is the total length of queried messages), compared to efficiency rate 1 and security $\mu^2/2^n$ achieved by CBC.

As another option, which is what is done in practice, one could construct the compression function via the Davies-Meyers transform $h(x, y) = E_x(y) \oplus y$. For one thing,

[1] Their attack was specific to a two-block CBC, but it is not hard to extend it to more rounds.

this is not very efficient, as it requires one to re-key the block cipher for every call, which is quite expensive for current block ciphers. (For example, eliminating this inefficiency was explicitly addressed and left as a challenge by Black, Cochran and Shrimpton [9].) More importantly, however, using the Davies-Meyers construction requires very strong assumptions on the block cipher to prove security. Namely, one can either make an ad hoc assumption that the Davies-Meyers compression function satisfies the needed domain extension property (such as being a PRF or a MAC), or formally prove the security of the construction in the ideal cipher model. Both of these options are unsatisfactory. The first option is provably not substantiated even is the block cipher E is assumed to be a pseudorandom permutation (PRP): for example, one can construct (artificial) PRPs for which the Davies-Meyers construction is not even unpredictable. As for the second option, it might be acceptable when dealing with strong properties, like collision-resistance or indifferentiability, when it is clear that the basic PRP property of the block cipher will not be enough [25]. However, to get pseudorandomness, or even unpredictability, going through the ideal cipher argument seems like a very (and unnecessarily) heavy hammer.

NEW MODE OF OPERATION. The above deficiencies of the CBC and the MD mode suggest that there might be a need to design a new mode of operation based on block ciphers, or, more generally, length-preserving (keyed or unkeyed) functions. We propose such a mode which will satisfy the following desirable properties:

- The mode is efficient. If the message length is ℓ blocks, we evaluate the block cipher at most $c\ell$ times (c is called the *efficiency rate*; we will achieve $c = 2$).
- The mode uses a small, constant number of (secret or public, depending on the application) keys for the block cipher. In particular, one never has to re-key the block cipher with some a-priori unpredictable value.
- It gives a provably secure VIL-MAC from length-preserving FIL-MAC, answering Question 1.
- It gives a provably secure VIL-PRF from a length-preserving FIL-PRF, therefore providing the hedge against the security of the block cipher E: if E is pseudorandom, the mode gives a PRF; if E is only unpredictable, one at least gets a MAC.
- It gives a way to build a VIL-RO (and, hence, a VIL-CRHF) from several random permutations.
- The mode is elegant and simple to describe.

Of course, simply being a "secure" domain extension for PRF/MAC/RO is not enough: the exact security achieved by the reduction is a crucial parameter, and we will elaborate on this later in this section.

ENCIPHERED CBC. The mode, *enciphered CBC*, we present in this paper is a relatively simple variant of the CBC mode. We first describe our "basic" mode, which works for domain-extension of MACs, PRFs and ROs, and later show the changes needed to make it work with (random) permutations as well.[2] The basic mode, depicted in

[2] In the random permutation model (where there are no secret keys) we need to worry about the inverse queries of the attacker. In contrast, in the secret key setting, a PRF is also a PRP, so the simpler mode already works for the domain extension of MACs and PRFs.

Figure 1, consists of three independent length-preserving functions f_1, f_2, f_3 (either keyed or not, depending on whether we are in the secret key setting, or in the random oracle model). First, we define an auxiliary compression function $g(x, y) = f_1(x) \oplus f_2(y)$. Intuitively, the key property of this function — which will hold in all our applications — is that it is weakly collision-resistant (WCR) [1]. This means that, given oracle access to f_1 and f_2, it is infeasible to find a collision for g. Then we use $g(x, y)$ as the compression function in the usual MD mode with strengthening: namely, we apply the Merkle-Damgård chaining to the message $(x_1 \dots x_\ell, \langle \ell \rangle)$, where $(x_1 \dots x_\ell)$ is our original message, and get output z. Finally, we output $f_3(z)$ as the value of our (basic) enciphered CBC.[3]

As we argue, if f_1, f_2, f_3 are three independent (keyed) MACs, then the above construction is a (three-keyed) VIL-MAC, answering Question 1. Also, although about twice less efficient than CBC, enciphered CBC also preserves the PRF property. On the other hand, if f_1, f_2, f_3 are random oracles, then the construction is indifferentiable from a VIL-RO. Finally, if we *assume* that f_1 and f_2 are such that $g(x, y)$ above is collision-resistant, then the mode which outputs the value z (and not $f_3(z)$) above is trivially collision resistant, since this is simply the usual MD transform with strengthening applied to a FIL-CRHF. Thus, if f_3 is "collision-resistant" (either trivially if it is a permutation, or even computationally), then enciphered CBC gives a VIL-CRHF. Of course, the assumption on g is not entirely satisfactory, but we argue that it is meaningful in the standard model.

OPTIMIZATIONS. We also show several optimizations of our mode which, while slightly less efficient, also work for two, or even one length-preserving round function. We only mention the two-key mode, since the one-key mode is a bit less "elegant" and intuitive to describe. The solution we propose (using two functions f and f') is to view $\{0, 1\}^n$ as the finite field $\mathbb{GF}(2^n)$, and then use the three-key solution with functions $f_1(x) = f(x)$, $f_2(y) = \alpha \cdot f(y)$ and $f_3(z) = f'(z)$, where α is any constant in $\mathbb{GF}(2^n)$ different from 0 and 1.[4] Then, we show that the resulting function $g(x, y) = f(x) \oplus \alpha \cdot f(y)$ is still WCR in all our applications.

Finally, we show how to extent the basic enciphered CBC mode to the case of random permutations. As already mentioned in Footnote 2, this is only the issue in the results concerning the random permutation model, since there the attacker can try to invert the random permutation. Indeed, the function $g(x, y) = f_1(x) \oplus f_2(y)$ is obviously *not* collision-resistant (which is crucial for our proof) if the attacker can invert f_1 or f_2. Our solution is to use the Davies-Meyers transform, but without the key. Namely, if π_1 and π_2 are random permutations, we essentially apply the previous mode to functions $f_1(x) = \pi_1(x) \oplus x$ and $f_2(y) = \pi_2(y) \oplus y$. This ensures that the function $g(x, y) = \pi_1(x) \oplus x \oplus \pi_2(y) \oplus y$ is still WCR, even with the oracle access to π_1^{-1} and π_2^{-1}. As for the function f_3, it really must look like a random oracle, so we use a slightly more

[3] One can also describe enciphered CBC as "enciphering" the input and the output of the standard CBC mode applied to f_1: we encipher all the input blocks (except the first) with f_2, and the output block — with f_3. This (less useful) view explains the name of the mode.

[4] We recommend the constant corresponding the the "polynomial" X in $\mathbb{GF}(2^n)$, since multiplication by this polynomial in $\mathbb{GF}(2^n)$ corresponds to one right shift and one XOR (the latter only if there is a carry), which is very efficient.

involved construction $f_3(z) = \pi_3(z) \oplus \pi_3^{-1}(z)$.[5] With these definitions of f_1, f_2 and f_3 using π_1, π_2 and π_3, we get our final enciphered CBC mode on block ciphers. (As we mentioned, though, the simplified mode already works for the case of PRFs and MACs.) We believe that optimizations similar to those made to the simplified mode, might also reduce the number of random permutations below three, but we leave this question to future work.

SECURITY. We will now discuss how the security of our mode for MAC/PRF/RO compares to known constructions. Recall that a mode of operation has rate c if it makes $c\ell$ calls to the underlying primitive when given an ℓ-block message. We achieve $c = 2$.

We will say that a domain extension for MACs has security d, if the security of the mode is $\epsilon \cdot \mu^d$ where ϵ is the security of the underlying FIL MAC and μ denotes the total length of the messages an adversary is allowed to query. Our mode achieves security $d = 4$, and this is the first constant-rate construction to achieve any security at all. For *shrinking* MACs $\{0,1\}^{n+k} \rightarrow \{0,1\}^n$, An and Bellare [1] show that a version of Merkle-Damgård gives a secure domain extension with security $d = 2$ (and rate $c = n/k$, which is constant if $k = \Omega(n)$). This security is much better than what we achieve, but it is unclear how to build a shrinking MACs with good security and compression efficiency (i.e., $k = \Omega(n)$) from a length-preserving MAC. Indeed, prior to this work, the best known construction of Dodis and Puniya builds a shrinking MAC with rate $c = \omega(\log \lambda)$ (where λ is the security parameters) and security $d = 6$, which is inferior to our $c = 2$ and $d = 4$.

As for PRFs, our mode achieves basically the same security $\mu^2/2^n$ as encrypted CBC, which is the best security known for constructions which are iterated, stateless and deterministic. In fact, as discussed in Section 3.3,we will achieve even better exact security when using PRPs (i.e., block ciphers) in place of length-preserving PRFs.

Similarly to MACs, we will say that a construction of a VIL-RO has security d, if it is $\mu^d/2^n$ indifferentiable from a random oracle when instantiated with FIL-ROs or RPs. With this convention, our construction has security $d = 4$. Recently, Maurer and Tessaro [17] give a pretty involved construction with the optimal security rate $d \rightarrow 1$ (at the expense of large efficiency rate $c = O(1)$), while the results of Coron et al. [10] for domain extension of "shrinking ROs" easily imply (by chopping some output bits of the length-preserving RO) a range of constructions with efficiency c and security $\mu^2/2^{(1-1/c)n}$. Although approaching security $d = 2$ for a large constant c, for $c = 2$ this gives poorer security $\mu^2/2^{n/2}$ than the security $\mu^4/2^n$ of enciphered CBC.

In the context of building VIL-CRHFs from length-preserving ROs or RPs, Shrimpton and Stam [24] give a simple construction from ROs with $c = 3$ and optimal $d \approx 2$, while Rogaway and Steinberger [23] recently reported a more complicated construction from RPs with $c = 3$ and optimal $d \approx 2$. Additionally, in a companion paper [22] they showed the necessity of non-trivial efficiency/security tradeoffs for any construction of VIL-CRHF in the random permutation model. This suggests the existence of similar (or worse) tradeoffs for the related question of building VIL-RO from length-preserving FIL-RO (or RP).

[5] This construction is of independent interest since it shows an indifferentiable construction of an n-to-n-bit random oracle from an n-to-n bit random permutation.

To summarize this discussion, we designed the first mode of operation *simultaneously* satisfying several demanding properties, some of which were never satisfied before (even in isolation). We conjecture that *any such mode must require some non-trivial tradeoff between efficiency and security*. Our specific mode, while simple and elegant, might not give such optimal tradeoffs. In particular, its security of "only" $\mu^4/2^n$ for the case or ROs and $\epsilon \cdot \mu^4$ for MACs is particularly unsatisfying to make it useful in practice (where $n = 128$; note that $\epsilon \geq 2^{-n}$). It is an interesting open question to understand the optimal efficiency/security tradeoffs, and to potentially improve upon our specific enciphered CBC mode of operation.

2 Preliminaries

We assume that the reader is familiar with the basic security definitions for MACs, PRFs, CRHFs and indifferentiability from RO. We use exact security definitions for each of these primitives.

MACs and PRFs. The security of a MAC is measured via its resistance to existential forgery under *chosen message attack* (see [3]). A function family F is a (t, q, μ, ϵ)-secure MAC if the success probability of any attacker with running time t, number of queries q and total message length μ is at most ϵ. Similarly, the security of PRFs is measured in terms of its indistinguishability from a truly random function under a chosen message attack, and a (t, q, μ, ϵ)-secure PRF is similarly defined.

Indifferentiability from Random Oracle. We follow the definitions of [10] for indifferentiability of a construction from an ideal primitive $\mathcal{F}$ (which will be a random oracle in this paper). A construction C, that has oracle access to ideal primitive $\mathcal{G}$, is $(t_D, t_S, q, \mu, \epsilon)$-indifferentiable from another ideal primitive $\mathcal{F}$, if there is a $\mathcal{G}$ simulator S that runs in time at most t_S, such that any attacker D with running time t_D, number of queries q and total query length μ can distinguish the $\mathcal{F}$ model (with access to $\mathcal{F}$ and S) from the $\mathcal{G}$ model (with access to C and $\mathcal{G}$) with advantage at most ϵ.

Collision Resistance. A function family F is (t, ϵ)-secure CRHF family, if the advantage of any attacker running in time t to find a collision for an f sampled at random from F, is at most ϵ.

3 Three-Key Enciphered CBC Construction

In this section, we will define the three-key enciphered CBC mode of operation and analyze its security under various notions.

First, we make some auxiliary definitions. Given two length-preserving functions $f_1, f_2 : \{0,1\}^n \rightarrow \{0,1\}^n$, we define the shrinking **XOR compression function**, $g[f_1, f_2]$, from $2n$ bits to n bits by $g[f_1, f_2](x_1 \parallel x_2) \stackrel{\text{def}}{=} f_1(x_1) \oplus f_2(x_2)$, where $x_1, x_2 \in \{0,1\}^n$. Given this function, we define the **XOR hash function** $G[f_1, f_2]$ to be simply the cascade construction applied to the XOR compression function. Namely, given input $x = x_1 \parallel \ldots \parallel x_\ell$, where $x_i \in \{0,1\}^n$, we let

$$G[f_1, f_2](x_1 \parallel \ldots \parallel x_\ell) \stackrel{\text{def}}{=} g[f_1, f_2](x_\ell \parallel g[f_1, f_2](\ldots g[f_1, f_2](x_2 \parallel x_1) \ldots))$$

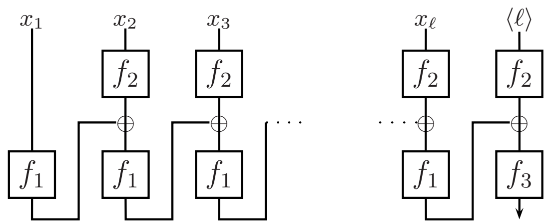


Fig. 1. The basic three-key enciphered CBC construction $H[f_1, f_2, f_3]$

THE CONSTRUCTION. The new mode of operation, $H[f_1, f_2, f_3]$, uses three length-preserving functions $f_1, f_2, f_3 : \{0,1\}^n \rightarrow \{0,1\}^n$ and takes a variable-length input $x = x_1 \parallel \ldots \parallel x_\ell$ (wlog, we assume the length to be a multiple of n; if not, then a suitable encoding scheme can be used to achieve this, such as appending a 1 followed by 0s). It simply applies the XOR hash function $G[f_1, f_2]$ described above to a suffix-free encoding of the input, followed by the third length-preserving function f_3. The particular suffix-free encoding we use is *Merkle-Damgård (MD) strengthening* [11,18], where one simply appends the input length $\langle \ell \rangle$ to the input. The resulting mode, depicted in Figure 1, is called *enciphered CBC mode*, and it is defined as:

$$H[f_1, f_2, f_3](x_1 \parallel \ldots \parallel x_\ell) = f_3(G[f_1, f_2](x_1 \parallel \ldots \parallel x_\ell \parallel \langle \ell \rangle))$$

3.1 VIL-MAC from Length-Preserving FIL-MAC

In this section we will prove, that unlike plain CBC, the enciphered CBC (cf. Figure 1) does give a secure VIL-MAC when instantiated with a length preserving MACs (here denoted $f_{k_1}, f_{k_2}, f_{k_3}$ to emphasize the secret keys k_1, k_2, k_3). We will use an elegant methodology of An and Bellare [1] which they used to analyze their *NI Construction* of a VIL-MAC from a shrinking FIL-MAC. However, we will see that it will be useful in our setting as well. In brief, the methodology introduced a notion of *weak collision-resistance* (WCR) and essentially reduced the construction of a VIL-MAC to that of a FIL-WCR. Details follow.

WEAK COLLISION-RESISTANCE (WCR). Consider a keys family of functions $F = \{f_k\}$, and the following attack game involving this function family. An attacker A gets oracle access to f_k (for random k) and returns a pair of messages $m \neq m'$ in the domain of F. The attacker A wins if these message collide: $f_k(m) = f_k(m')$. The function family F is said to be a (t, q, μ, ε)-*secure WCR* function family if the success probability of any attacker with running time t, number of queries q and total message length μ is at most ε.

FROM WCR TO MAC. The methodology of An and Bellare [1] utilized the notion of WCR via the following reasoning (which we immediately attempt to apply to the case of enciphered CBC).

Step 1: The composition of a FIL-MAC f_k and a WCR function $h_{k'}$ is a secure MAC $f_k(h_{k'}(\cdot))$ (Lemma 4.2 [1]). Applied to enciphered CBC, where f_{k_3} is a FIL-MAC, it means that it suffices to show that the XOR hash function $G[f_{k_1}, f_{k_2}]$, with suffix-free inputs, is a VIL-WCR.

Step 2: The cascade construction, with suffix-free inputs, applied to a FIL-WCR function gives a VIL-WCR function (Lemma 4.3 [1]). In our case, the XOR hash function is exactly the required cascade construction applied to the XOR compression function $g[f_{k_1}, f_{k_2}]$. Thus, it suffices to show that the latter is FIL-WCR.

Step 3: Build a FIL-WCR. In the case of the NI Construction of [1], one needed to build a FIL-WCR from a shrinking MAC, which was easy to do: any shrinking FIL-MAC is FIL-WCR (Lemma 4.4 [1]). Applied to our setting, it would suffice to show that the XOR compression function $f_{k_1}(x_1) \oplus f_{k_2}(x_2)$ is a FIL-MAC. However, this is easily seen to be false: for example, the XOR of its outputs applied to inputs $(x_1 \parallel x_2)$, $(x_1 \parallel x_2')$, $(x_1' \parallel x_2)$ and $(x_1' \parallel x_2')$ is always 0^n, which easily leads to a forgery. Despite this "setback", we give a direct proof that the XOR compression function is a FIL-WCR, despite not being a FIL-MAC. And this is all we need.

Lemma 1. *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions. Define the function family $g[f_{k_1}, f_{k_2}](x_1 \parallel x_2) \stackrel{def}{=} f_{k_1}(x_1) \oplus f_{k_2}(x_2)$. If the function family f is a (t, q, qn, ϵ)-secure MAC family, then $g[f_{k_1}, f_{k_2}]$ is a $(t', q, 2qn, \epsilon \cdot q^4/2)$-secure WCR family, where $t' = t - O(qn)$.*

Proof: Let A be an adversary which finds a collision for $g[f_{k_1}, f_{k_2}]$ with probability ϵ' (if k_1, k_2 are uniformly random). From such an A we will construct a new adversary B which is basically as efficient as A, and which forges f with probability at least $2\epsilon'/q^4$. Instead only giving A access to $g[f_{k_1}, f_{k_2}]$, we allow A to make q queries to f_{k_1} and f_{k_2} respectively, but we require this queries are made alternately, i.e. after a query to f_{k_b}, A must make a query to $f_{k_{3-b}}$ (note that such an A can trivially simulate q queries to $g[f_{k_1}, f_{k_2}]$). Moreover we assume that if $x_1 \| x_2, x_1' \| x_2'$ is A's final output, then A always made the f_{k_1} queries x_1, x_1' and the f_{k_2} queries x_2, x_2' (this can be done wlog. if we allow A two extra queries to f_{k_1} and f_{k_2} respectively). Assume A is successful, and finds a collision $x_1 \| x_2 \neq x_1' \| x_2'$ for $g[f_{k_1}, f_{k_2}]$. We say that a query x (say to f_{k_1}) is a winner query, if it is the first query where for some b, c, d, the pair $x \| b \neq c \| d$ is a collision for $g[f_{k_1}, f_{k_2}]$ and A already knows (i.e. made the queries) $f_{k_2}(b), f_{k_1}(c), f_{k_2}(d)$. Note that if A found a collision, then it must have made a winner query. Our attacker B, which must forge f_k (for some random unknown k) is now defined as follows. First B flips a random coin $r \in \{1, 2\}$, and samples a random key k' for f. Now B lets A attack f_{k_1}, f_{k_2}, where $f_k = f_{k_r}$ and $f_{k'} = f_{k_{3-r}}$. During the attack, for a random $i, 2 \leq i \leq q$, B stops when A makes the i'th query x to f_{k_r} and "guesses" that this will be the winning query. Then B randomly chooses three already made queries b, c, d, conditioned on $x \| b \neq c \| d$ (hoping that $x \| b, c \| d$ is a collision), and guesses the forgery $\rho := f_{k_{3-r}}(b) \oplus f_{k_r}(c) \oplus f_{k_{3-r}}(d)$ for $f_{k_r} = f_k$ for the message x. Note that ρ is a good forgery for $f_k = f_{k_r}$, if $x \| b, c \| d$ is indeed a collision for $g[f_{k_r}, f_{k_{3-r}}]$. Thus B will be successful if A makes a winning query (which happens with probability ϵ'), and moreover B correctly guesses r (i.e. whether the winning query will be a f_1 or f_2 query), the index i of the winning query and also the three other queries involved. The probability of all that guesses being correct is at least $2\epsilon'/q^4$.

By assumption (on the security of f as a MAC) we have $2\epsilon'/q^4 \leq \epsilon$, thus the success probability of B must be at most $\epsilon \cdot q^4/2$ as claimed. ☐

Combining this result with the Lemmas 4.2 and 4.3 from [1], we immediately get

Theorem 1. *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a (t, q, qn, ε)-secure length-preserving FIL-MAC. Then $H[f_{k_1}, f_{k_2}, f_{k_3}](.)$ (where k_1, k_2, k_3 is the secret key) is a $(t', q, qn, \varepsilon \cdot q^4)$-secure variable input-length MAC, where $t' = t - O(qn)$.*

3.2 VIL-RO from Length-Preserving FIL-RO

In this section we show that the enciphered CBC mode provides a domain extension for length-preserving ROs (in the sense of [10]).

Theorem 2. *Consider three length-preserving ROs $f_1, f_2, f_3 : \{0,1\}^n \rightarrow \{0,1\}^n$. Then the enciphered CBC construction $H[f_1, f_2, f_3]$ is $(t_D, t_S, q, \mu, \epsilon)$-indifferentiable from a VIL-RO. Here $t_S = \mathcal{O}(q^2)$, $\epsilon = \mathcal{O}((q+\mu)^4/2^n)$ and t_D is arbitrary.*

One might hope that the proof of this theorem can be given by using the corresponding indifferentiability result of Coron et al [10] for the NMAC construction. However, this intuition turns out to be incorrect since in order to use the result of [10], we will need to show that the *XOR compression function* $g[f_1, f_2]$ is indifferentiable from a FIL-RO from $2n$ bits to n bits. But this is clearly false, since for three n-bit input blocks x, y, y', we can see that $g[f_1, f_2](x \parallel y) \oplus g[f_1, f_2](x \parallel y')$ is independent of the n-bit block x which is certainly not true for an ideal FIL-RO!

Hence we give a direct proof for this result. In the proof, we need to construct a FIL-RO simulator that responds to the queries made by the indifferentiability attacker A to the FIL-ROs f_1, f_2 and f_3 in the VIL-RO model. Roughly speaking, the simulator responds to f_1 and f_2 queries at random and hopes that no collisions occur for the input to f_3 in the last round of the enciphered CBC construction. If no such collisions occur, then it can adjust its response to f_3 queries to match the VIL-RO output on the corresponding variable-length input (which it finds by searching through its previous responses).

Proof: We will prove the indifferentiability of the enciphered CBC mode of operation $H[f_1, f_2, f_3]$ from a variable input-length random oracle (VIL-RO) $F : \{0,1\}^* \rightarrow \{0,1\}^n$, in the random oracle model for the underlying fixed input-length functions $f_1, f_2, f_3 : \{0,1\}^n \rightarrow \{0,1\}^n$. The proof consists of two parts: the description of the FIL-RO simulator and the proof of indifferentiability.

The Simulator. The simulator S responds to queries of the form (i, x), where $i \in \{1, 2, 3\}$ and $x \in \{0,1\}^n$. In particular, the response $y \in \{0,1\}^n$ of the simulator S to a query (i, x) will be interpreted as the output $f_i(x)$ by the distinguisher, i.e. $y = f_i(x)$. The simulator also maintains a table $\mathcal{T}$ consisting of entries of the form (i, x, y), for each query (i, x) that it responded to with the output y.

f_1 QUERIES. In response to a query of the form $(1, x)$, the simulator S looks up its table for an entry of the form $(1, x, y)$. If it finds such an entry, then it responds with the output y recorded in this tuple, otherwise it responds to this query by choosing an output y that is uniformly distributed over $\{0,1\}^n$ and records the tuple $(1, x, y)$ in its table $\mathcal{T}$.

f_2 QUERIES. The simulator responds to queries of the form $(2, x)$ in the same way as it responds to f_1 queries, i.e. first looking up its table for a matching tuple $(2, x, y)$, else responding with a fresh uniformly distributed output y.

f_3 QUERIES. In response to queries of the form $(3, x)$, the simulator needs to check if there is a variable length input X, such that it needs to be consistent with the VIL-RO output $F(X)$ on this input. It firsts looks up its table $\mathcal{T}$ to find out if there is a matching tuple $(3, x, y)$ corresponding to a duplicate query, in which case it responds with y. Otherwise, it looks up the table $\mathcal{T}$ for a sequence of tuples $(1, x_1^1, y_1^1) \ldots (1, x_i^1, y_i^1)$ and $(2, x_1^2, y_1^2) \ldots (2, x_i^2, y_i^2)$, that satisfy the following conditions:

(a) For $j = 2 \ldots i$, it holds that $x_j^1 = y_{j-1}^1 \oplus y_{j-1}^2$.
(b) For the last tuples $(1, x_i^1, y_i^1)$ and $(2, x_i^2, y_i^2)$, it holds that the current f_3 input $x = y_i^1 \oplus y_i^2$.
(c) The bit string $x_1^1 \parallel x_1^2 \parallel \ldots \parallel x_i^2$ is such that $x_i^2 = \langle i \rangle$. That is, it should be the output of Merkle-Damgård strengthening applied to a valid input.

If the simulator finds such a sequence of tuples, then it queries the VIL-RO F to find out the output $y = F(x_1^1 \parallel x_1^2 \parallel \ldots \parallel x_{i-1}^2)$ and responds to the query $(3, x)$ with the output y, and records the tuple $(3, x, y)$ in its table $\mathcal{T}$. If it does not find such a sequence of tuples then it responds with a uniformly random output $y \in \{0,1\}^n$ and records $(3, x, y)$ in $\mathcal{T}$.

The proof of indifferentiability is postponed to the full version of this paper [13]. ◻

3.3 VIL-PRF from Length-Preserving FIL-PRF

If we remove the f_2 boxes in our enciphered CBC mode of operation (cf. Figure 1), we get a well known mode of operation called *encrypted CBC*, which is known to be a good domain extension for PRFs [19,20]. The security of encrypted CBC (i.e. the distinguishing advantage from a uniformly random function, URF) when instantiated with two PRFs is $(\mu^2/2^n + 2\epsilon)$, where μ is the total length (in n bit blocks) of the messages queried and ϵ is a term that accounts for the insecurity of the underlying PRF. It is not surprising that our enciphered CBC mode is almost as secure, as the application of f_2 (not present in the usual encrypted CBC mode) does not affect the security by much.

Theorem 3. *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a $(t, \mu, \mu n, \epsilon)$-secure FIL-PRF family. Then $H[f_{k_1}, f_{k_2}, f_{k_3}](.)$ is a $(t', q, \mu n, 2\mu^2/2^n + 3\epsilon)$-secure VIL-PRF family where $t' = t - O(qn)$.*

We will not formally prove this theorem, but just explain how it follows from the known $(t', q, \mu n, \mu^2/2^n + 2\epsilon)$ security of the encrypted CBC-MAC (under the same assumption on the PRF like in the theorem). The main observation here is that we can turn any distinguisher D for enciphered CBC into a distinguisher D' for encrypted CBC, by simply sampling some key k_2 at random, and then enciphering with f_{k_2} (except the first block) the queries made by D, before forwarding them to the oracle of D'. If the oracle of D' is *encrypted* CBC, then the oracle's answers look *exactly* as if they were computed by an *enciphered* CBC. In the ideal experiment, where the oracle of D' is a VIL-URF, the oracle's answers still look uniformly random, even if the input is first applied to f_{k_2}, unless D makes two queries containing blocks $x \neq x'$ which collide on f_{k_2}. The probability of that happening can be upper bounded by $\mu^2/2^n + \epsilon$, as f_{k_2} can be distinguished from a URF with advantage at most ϵ, and the probability to find a collision for a URF with range $\{0,1\}^n$ making μ queries is at most $\mu^2/2^n$. This $\mu^2/2^n + \epsilon$ accounts for the gap in the security for enciphered CBC (as in the theorem) and encrypted CBC (as mentioned above).

IMPROVING THE BOUND FOR BLOCK CIPHERS. As just explained, the gap in the security of encrypted and enciphered CBC is bounded by the probability that one can find a collision for the PRF f_{k_2}. Thus, if f_{k_2} is a permutation (where there are no collisions), $(t, q, \mu n, \delta)$-security for encrypted CBC implies basically the same $(t - O(\mu n), q, \mu n, \delta)$ security for enciphered CBC. This observation is useful, as in practice the PRF is usually instantiated by a block cipher, which is a permutation. And further, for the encrypted CBC mode of operation, one can prove much better bounds than $(\mu^2/2^n + 2\epsilon)$ if both f_{k_1} and f_{k_2} are assumed to be pseudorandom permutations (PRPs) [4,20] as opposed to PRFs. Thus, this better bounds for encrypted CBC translate directly to our mode of operation. To state the improved bounds, one must assume an upper bound ℓ on the length of *each* message queried by the distinguisher (this should not be a problem in practice, as the bound can be exponential). Let q be the number of queries the adversary is allowed to make, then if no messages is longer than $\ell \leq 2^{n/4}$ (and thus the total length μ is at most ℓq), the security of encrypted CBC instantiated with PRPs is $q^2\ell^{\Theta(1/\ln\ln\ell)}/2^n$ (plus some ϵ term accounting for the insecurity of the PRP). With the stronger condition that $\ell \leq 2^{n/8}$, one gets an even stronger $O(q^2/2^n)$ bound [20], which is tight up to a constant factor. Note that this is much better than the $O(q^2\ell^2/2^n)$ bound implied by Theorem 3, and in particular is independent of the message length ℓ.

3.4 Collision Resistance of Enciphered CBC

Now we discuss the collision-resistance of the enciphered CBC mode of operation. Note that the problem of constructing variable input-length CRHFs from length-preserving collision-resistant (CR) functions does not make much sense, since it is trivial to construct length-preserving CR functions (such as the identity function). However, as discussed in the introduction, we can make the following simple observation about the enciphered CBC mode of operation.

Lemma 2. *Consider three length-preserving functions f_1, f_2 and f_3 on n bits. If the XOR compression function $g[f_1, f_2]$ and the function f_3 are collision-resistant, then the enciphered CBC mode of operation, $H[f_1, f_2, f_3]$, is collision-resistant as well.*

This observation is a simple consequence of the result of Merkle-Damgård [11,18], since we already use a suffix-free encoding the the enciphered CBC mode. Notice that assuming that a length-preserving function f_3 is a CRHF is a very mild requirement, since any permutation trivially satisfies this property. Thus, we the main assumption we need is that the XOR of functions f_1 and f_2 is a CRHF. Of course, in the random oracle model, it is well known the the XOR of two random oracles is collision-resistant (in fact, in this setting we showed in Section 3.2 that the enciphered CBC mode even gives a VIL-RO, let alone a "mere" VIL-CRHF).

Our point is that it is not essential to make idealized assumptions on the functions f_1 and f_2 to prove collision resistance of the construction $g[f_1, f_2]$. For instance, consider any finite field $\mathbb{F}$ for which the *discrete logarithm* problem is hard, and whose elements can be naturally encoded as binary strings. Define the functions $f_1, f_2 : \{0,1\}^n \to \{0,1\}^n$ as $f_1(x) = gen_1^x$ and $f_2(x) = gen_2^x$, where gen_1 and gen_2 are two generators of $\mathbb{F}$. Further, let us replace the XOR operation in $g[f_1, f_2]$ by a field-multiplication over $\mathbb{F}$. Then we get a new function $g(x \parallel y) = gen_1^x \cdot gen_2^y$ which is provably collision-resistant under the discrete log assumption. Coupled with the RO justification, this example suggests that our assumption on $g[f_1, f_2]$ is not too unreasonable.

We stress, though, that the XOR compression function is definitely *not* collision-resistant when f_1 and f_2 are (public) random *permutations*, as any two pairs (x, y), $(x', f_2^{-1}(f_1(x) \oplus f_2(y) \oplus f_1(x')))$ give a collision. Indeed, as we explain next, our mode has to be slightly modified to handle the case of random permutations.

4 A Block Cipher Based Mode of Operation

So far we described the enciphered CBC mode for three length-reserving functions. But, as already mentioned at the end of Section 3.4 and in Footnote 2, we need to modify our basic mode in order for it to work with permutations in "unkeyed" settings, such as indifferentiability from RO and collision-resistance. In the "keyed" settings, i.e. for MACs and PRFs, replacing the functions with permutations does not make a qualitative difference (up to a birthday bound), since a PRP is also a PRF. Thus, the enciphered CBC construction works for domain extension of MACs and PRFs even if one uses a block cipher to implement the these primitives. However, even in these cases the construction may have slightly different (up to a birthday bound) exact security. For instance, as discussed for the case of PRFs in Section 3.3, the enciphered CBC construction has actually *better* exact security if permutations are used instead of functions.

"ENHANCED" ENCIPHERED CBC. We now described the (enhanced) enciphered CBC mode of operation based on three permutations π_1, π_2 and π_3. While this more complicated mode is only needed for the "unkeyed" settings (RO and CRHF), we will see that it still works for the "keyed" settings (PRF and MAC), although under slightly stronger assumptions than before. The mode is depicted in Figure 2 and is denoted $H^*[\pi_1, \pi_2, \pi_3]$. We observe that this enhanced mode is *precisely* the basic enciphered CBC construction $H[f_1, f_2, f_3]$ with length-preserving functions f_1, f_2 and f_3 defined as follows: $f_i(x) = \pi_i(x) \oplus x$ for $i = 1, 2$, and $f_3(x) = \pi_3(x) \oplus \pi_3^{-1}(x)$. The reason for this choice will become clear in the sequel, when we discuss why this "enhanced" mode works for building VIL-RO and VIL-CRHF.

4.1 Collision Resistance from Random Permutations

Using Lemma 2, in order to argue the collision-resistance of the enhanced mode, it suffices to argue the collision resistance of the XOR compression function

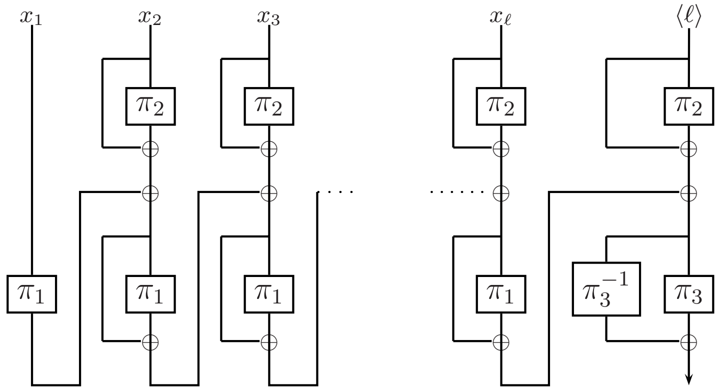


Fig. 2. The "enhanced" three-key enciphered CBC construction $H^*[\pi_1, \pi_2, \pi_3]$ which is a domain extender for random oracles, even if instantiated with random *permutations*

$f_(x) \oplus f_2(y) = \pi_1(x) \oplus x \oplus \pi_2(y) \oplus y$, and the function $f_3(x) = \pi_3(x) \oplus \pi_3^{-1}(x)$, even if the attacker can invert π_1, π_2 and π_3. In the standard model, we will have to simply make these (unusual but not unreasonable) assumptions for whatever public permutations we end up using. However, we must first justify that these assumption at least hold in the random permutation model. We start with the XOR compression function.

Lemma 3. *For two independent permutations π_1, π_2, the XOR compression function $g[f_1, f_2]$ (with f_1 and f_2 as defined above) is (t, ϵ)-collision-resistant in the random permutation model for π_1 and π_2. Here $\epsilon = q^4/2^n$ if the attacker makes at most $q \leq \min(t, 2^{n-1})$ random permutation queries.*

Proof: Let A be any collision-finding attacker who outputs a collision $(x_1 \parallel x_2), (x_1' \parallel x_2')$. When the attacker makes its forward query x to π_i (here $i = 1, 2$) or a backward query y to π_i^{-1}, we will record a tuple $(i, x, pi_i(x))$ or $(i, pi_i^{-1}(y), y)$ to a special table T. Wlog, we assume that A does not make redundant queries and that, at the the end of the game, T contains all the "collision-relevant" values $(1, x_1, y_1 = \pi_1(x_1)$, $(1, x_1', y_1' = \pi_1(x_1')$, $(2, x_2, y_2 = \pi_2(x_2)$, $(2, x_2', y_2' = \pi_1(x_2')$. This means that instead of having A output a collision, we can declare A victorious if T contains 4 (not necessarily distinct) tuples, as above, such that $x_1 \oplus y_1 \oplus x_2 \oplus y_2 = x_1' \oplus y_1' \oplus x_2' \oplus y_2'$. To complete the proof, we will argue, by induction on $0 \leq j \leq q$, that after A makes his first j queries, the probability that T will contain the required 4-tuple is at most $j^4/2^n$.

Consider query number $j+1$. Wlog, assume it is to π_1 or π_1^{-1}. Then, either T already contained the colliding 4-tuple before this query was made (which, by induction, happens with probability at most $j^4/2^n$), or the answer to the current query $j+1$, together with 3 prior queries, resulted in the colliding equation. Let us fix any one of these at most j^3 choices of 3 prior queries. Once this choice is fixed, it defines a unique answer to query $j+1$ which will result in collision. Indeed, if the query $j+1$ is to $\pi_1(x_1)$, and the 3 prior table values are $(1, x_1', y_1')$, $(2, x_2, y_2)$, $(2, x_2', y_2')$, then the only answer y_1 which will result in collision is equal to $y_1 = x_1 \oplus x_1' \oplus y_1' \oplus x_2 \oplus y_2 \oplus x_2' \oplus y_2'$. Similarly, if the query was to $\pi_1^{-1}(y_1)$, then the only answer x_1 resulting in a collision is $x_1 = y_1 \oplus x_1' \oplus y_1' \oplus x_2 \oplus y_2 \oplus x_2' \oplus y_2'$. However, since the total number of queries $j \leq 2^{n-1}$, for each fresh query there are at least $2^n - j \geq 2^{n-1}$ equally likely answers. Thus, the chance that a random such answer will "connect" with a given subset of 3 prior queries is at most $1/2^{n-1}$.

Overall, the get that the probability that there will be a collision in T after $j+1$ queries is at most $j^4/2^n + j^3/2^{n-1} < (j+1)^4/2^n$, completing the proof. ◻

Next, we need to prove the collision resistance of the construction $f_3(x) = \pi_3(x) \oplus \pi_3^{-1}(x)$ in the random permutation model. However, this will trivially follow from a much stronger result we prove in the upcoming Lemma 4, which will be needed to prove the indifferentiability of our mode from a VIL-RO.

4.2 Building VIL-RO from Random Permutations

In this section we argue that the enhanced enciphered CBC mode gives a VIL-RO in the *random permutation* model for π_1, π_2, π_3. The actual proof (and the exact security) of this result is quite similar to the proof of Theorem 2. Therefore, instead of repeating the (long) proof of this result, we will only (semi-informally) highlight the key new ingredients of the proof which we must address in the random permutation model. Concentrating on these ingredients will also help us to "de-mystify" why we defined the functions f_1, f_2, f_3 in the way we did.

RANDOM ORACLE FROM RANDOM PERMUTATION. The most modular way to extend Theorem 2 to the random permutation model would be to show how to implement (in the indifferentiability framework) a length-preserving RO from an RP, and then use the general composition theorem in the indifferentiability framework (see [10]). And, indeed, it turns out that this is precisely what we did for the function f_3 (but *not* f_1 and f_2; stay tuned) by defining it as $\pi_3 \oplus \pi_3^{-1}$. Intuitively, f_3 must really look like a full-fledged FIL-RO in the proof of Theorem 2. The security of this construction for f_3 is of independent interest, since it builds a FIL-RO from an RP, and follows from the following Lemma (which also implies that f_3 is collision-resistant in the random permutation model):

Lemma 4. *Let $\pi : \{0,1\}^n \rightarrow \{0,1\}^n$ be a permutation. Then the construction $f[\pi] \stackrel{def}{=} \pi \oplus \pi^{-1}$ is $(t_D, t_S, q, \mu, \mathcal{O}(q^2/2^n))$-indifferentiable from a length-preserving FIL-RO on n bits in the* random permutation model *for π (here t_D is arbitrary and $t_S = \mathcal{O}(qn)$).*

Proof: We will show that the construction $f[\pi]$ is indifferentiable from a FIL-RO $F : \{0,1\}^n \rightarrow \{0,1\}^n$ in the random permutation model for $\pi : \{0,1\}^n \rightarrow \{0,1\}^n$. The proof consists of two parts: a description of the RP simulator S and the proof of indifferentiability.

The Simulator. The simulator S responds to queries of the form (i, x), for $i = -1, +1$ and $x \in \{0,1\}^n$. The distinguisher interprets the response of the simulator to a query $(+1, x)$ (resp. $(-1, x)$) as the (resp. inverse) permutation output $\pi(x)$ (resp. $\pi^{-1}(x)$). The simulator maintains a table $\mathcal{T}$ of permutation input-output pairs (x, y) such that, either it responded with y to a query $(+1, x)$ or with x to a query $(-1, y)$. On a query

$(+1, x)$ (resp. $(-1, y)$), S first searches its table $\mathcal{T}$ for a pair (x, y') (resp. (x', y)) and if it finds such a pair then it responds with y' (resp. y).

On a new query $(+1, x)$, the simulator searches its table for a pair of the form (x', x) (i.e. x was an earlier RP output). If it finds such a pair, then it queries the FIL-RO F to find the output $F(x)$. It then responds with the output $y = x' \oplus F(x)$, and records the pair (x, y) in its table $\mathcal{T}$.

On a new query $(-1, y)$, the simulator searches its table for a pair of the form (y, y') (i.e. y was an earlier RP input). If it finds such a pair, then it queries the FIL-RO F to find the output $F(y)$. It then responds with the $x = y' \oplus F(x)$ to the query, and records the pair (x, y) in its table $\mathcal{T}$.

The proof of indifferentiability is postponed to the full version of this paper [13]. ☐

Of course, we could have the above Lemma to define f_1 and f_2 as well, but this would double the efficiency rate of our enhanced mode from 2 to 4. Instead, we observe that in the proof of Theorem 2, we "only" need the functions f_1 and f_2 to be such that the XOR compression function $g[f_1, f_2]$ is what we call *extractable*.[6]

EXTRACTABILITY. Informally, a hash function g^f built from some oracle f is ϵ-extractable (where ϵ could depend on some other parameters), if there exists and extractor Ext such that no attacker A can "fool" Ext with probability more than ϵ in the following game. A is given oracle access to f and outputs a value y. Ext takes y and the oracle queries that A made to f so far, and attempts to outputs a preimage x of y under g^f. Then A is allowed to run some more (making more calls to f) and outputs its own preimage x' of y. Then A "fools" Ext if $g^f(x') = y$ but $x \neq x'$.

Coming back to our situation, where $f = (f_1, f_2)$ and $g^f = g[f_1, f_2](x_1 \parallel x_2) = f_1(x_1) \oplus f_2(x_2)$, we only need to argue the extractability of this construction in the random permutation model, when we define $f_i(x) = \pi_i(x) \oplus x$. The extractor for this construction is defined naturally: given y, search the list of A's queries for a pair of inputs/outputs (x_1, y_1), (x_2, y_2) to π_1 and π_2, respectively, such that $y = x_1 \oplus y_1 \oplus x_2 \oplus y_2$. If exactly one such pair is found, output $x = x_1 \parallel x_2$, else fail. The security of this extractor is given below.

Lemma 5. *For two independent permutations π_1, π_2, the XOR compression function $g[f_1, f_2]$ (with f_1 and f_2 as defined above) is extractable in the random permutation model for π_1 and π_2. In particular, if the attacker makes at most q permutation queries, it can fool the above extractor with probability at most $\mathcal{O}(q^4/2^n)$.*

We remark that extractability can be viewed as a slight strengthening of collision-resistance: indeed, finding a collision allows one to trivially fool any extractor with probability at least $1/2$. Not surprisingly, the proof of this Lemma is only marginally harder than the proof of Lemma 3. Omitting details, we use the proof of Lemma 3 to argue that the extractor will never find more than one preimage of y through A's oracle

[6] Technically, we need the whole XOR hash function $G[f_1, f_2]$ to be extractable, but it is easy to see that this is implied by the extractability of the compression function $g[f_1, f_2]$. In this case, if the XOR Hash function is extractable and the attacker makes an oracle call $f_3(y)$, the Simulator can extract the preimage $x = (x_1 \ldots x_\ell)$ of y and "define" $f_3(y) = F(y)$, where F is the VIL-RO.

queries. And if at most one such preimage is found, a similar argument can show that the chance of the attacker to find a different preimage x' of y is at most $q^2/2^n$.

This completes our high-level argument why the enhanced enciphered CBC mode yields a VIL-RO (and also explains our definition of f_1, f_2, f_3 in terms of π_1, π_2, π_3).

4.3 Revisiting Security for PRFs and MACs

Although the basic enciphered CBC mode already works for the case of PRFs and MACs, even when permutations are used, we argue that the enhanced mode continues to work for these settings as well. First, note that if π is a PRF (resp. MAC), then the construction $[\pi(x) \oplus x]$ is also a PRF (resp. MAC) with the same exact security. Thus, we do not need to make any stronger assumptions on π_1 and π_2 than what we made on f_1 and f_2. However, in order to prove that $f_3 = \pi_3 \oplus \pi_3^{-1}$ is a PRF (resp. MAC), we will need to make slightly stronger assumption on π_3 than being the "usual" PRF (resp. MAC). In some sense, this is expected since an inverse query to π_3 is used in the construction itself. Luckily, the extra assumptions we need are quite standard and widely believed to hold for current block ciphers. Specifically, for the case of PRFs we require that π_3 is a *(strong) pseudorandom permutation (sPRP)*: i.e., it remains a PRP even if the attacker can make both the forward and the inverse queries. Similarly, for the case of MACs, we need to assume that π_3 is a *(strong) unpredictable permutation (sUP)*: i.e., a permutation for which no attacker can produce a (non-trivial) forgery even if given oracle access to both the forward and the inverse queries. The proof of this simple lemma will be given in the full version.

Lemma 6. *Let $\Pi = \{\pi_k\}_k$ be a family of permutations, and define the family $F = \{f_k\}_k$ of length-preserving functions by $f_k(x) = \pi_k(x) \oplus \pi_k^{-1}(x)$. Then F is a*

- *$(t, q, \mu, \epsilon + \mathcal{O}(q^2/2^n))$-secure PRF if Π is a $(t + \mathcal{O}(qn), 2q, 2\mu, \epsilon)$-secure sPRP.*
- *$(t, q, \mu, \mathcal{O}(\epsilon \cdot q^2))$-secure MAC if Π is a $(t + \mathcal{O}(qn), 2q, 2\mu, \epsilon)$-secure sUP.*

of $\epsilon \cdot q^2$ might sound alarming, especially when combining this with the statement of Theorem 1, where there is an additional loss of the q^4 factor. However, a closer look at the proof of Theorem 1 reveals that the exact security of the enciphered CBC is actually at most $\epsilon_3 + (\epsilon_1 + \epsilon_2) \cdot q^4$, where ϵ_i is the security of f_i. Coupled with the above Lemma, we get security $\epsilon \cdot q^2 + (\epsilon_1 + \epsilon_2) \cdot q^4$ (where ϵ is the security of sUP π_3, and ϵ_1, ϵ_2 are the securities of MACs π_1 and π_2).

5 Two-Key Enciphered CBC Construction

In this section we show that it is possible to instantiate the (basic) enciphered CBC mode using only two independent length-preserving functions.

A first natural idea is to define the function f_2 in the three-key version using the function f_1. For example, we can make $f_2 = f_2$. However, in this case it is easy to see that the resulting mode is insecure for all the security notions considered in this paper. This is because the resulting XOR compression function $g[f_1, f_1]$ becomes a constant function 0^n on any "symmetric" input $(x \parallel x)$. Luckily, we show that this problem can be resolved by instantiating f_2 with a different multiple of f_1!

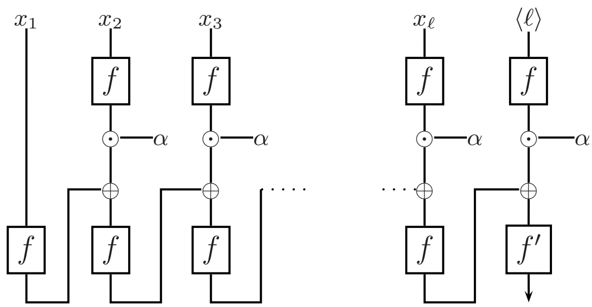


Fig. 3. The two-key enciphered CBC construction $H_\alpha[f, f']$

THE CONSTRUCTION. Consider a function $f : \{0,1\}^n \to \{0,1\}^n$. We can view the inputs/outputs of f as elements of the field $\mathbb{GF}(2^n)$, and the bit-by-bit XOR operation becomes addition over the field $\mathbb{GF}(2^n)$. Let α be any element of this field other than 0 or 1. Then we define the functions f_1 and f_2 in the enciphered CBC mode of operation $H[f_1, f_2, f_3]$ as follows: $f_2(\cdot) \stackrel{\text{def}}{=} f(\cdot)$ and $f_1(\cdot) \stackrel{\text{def}}{=} \alpha \cdot f(\cdot)$. We still use a different FIL function f' as the third function f_3 in the construction $H[f_1, f_2, f_3]$.

This defines the new XOR compression function $g_\alpha[f]$ as $g_\alpha[f](x_1 \parallel x_2) \stackrel{\text{def}}{=} f(x_1) \oplus (\alpha \cdot f(x_2))$. Intuitively, the key point we will repeatedly use in our analyses is that the function $g_\alpha[f]$ is still WCR (or even extractable in the RO model) when $\alpha \notin \{0,1\}$. We also denote the corresponding XOR hash function as $G_\alpha[f]$, and our new mode of operation using two functions f' and f' as $H_\alpha[f, f']$, where:

$$H_\alpha[f, f'](x_1 \parallel \ldots \parallel x_\ell) \stackrel{\text{def}}{=} f'\left(G_\alpha[f](x_1 \parallel \ldots \parallel x_\ell \parallel \langle \ell \rangle)\right)$$

The construction is illustrated in Figure 3. We will now analyze its security for various security notions.

5.1 Two-Key Enciphered CBC Is MAC Preserving

In the full version of the paper we prove the following lemma.

Lemma 7. *If the function family f is $(t, 2q, 2qn, \epsilon)$-secure MAC family, then $g_\alpha[f]$ is a $(t', q, 2qn, \epsilon \cdot 32 \cdot q^4)$-secure WCR family, where $t' = t - O(qn)$.*

As explained in Section 3.1, we can now use Lemma 7 along with Lemmas 4.2 and 4.3 from [1] to get the following Theorem.

Theorem 4. *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be a $(t, 2\mu, 2\mu n, \varepsilon)$-secure length-preserving FIL-MAC. Then $H_\alpha[f_k, f_{k'}](.)$ (where k, k' is the secret key) is a $(t', q, \mu n, \varepsilon \cdot 33 \cdot \mu^4)$-secure variable input-length MAC, where $t' = t - O(\mu n)$ and q is arbitrary.*

5.2 VIL-RO Using the Two-Key Construction

We now show that given two independent FIL-ROs $f, f' : \{0,1\}^n \to \{0,1\}^n$, the two-key enciphered CBC construction $H_\alpha[f, f']$ is indifferentiable from a VIL-RO F. The proof of indifferentiability for this construction is similar to the corresponding proof for the three FIL-RO enciphered CBC construction. The only difference is in the way the simulator searches for a variable length input where it might need to be consistent with the VIL-RO, when responding to a f' query. We give a proof of this lemma in the full version of this paper [13].

Theorem 5. *Consider two length-preserving functions* $f, f' : \{0,1\}^n \rightarrow \{0,1\}^n$*. Then the new enciphered CBC construction* $RO2[f, f']$ *is* $(t_D, t_S, q, \mu, \epsilon)$*-indifferentiable from a random oracle in the FIL-RO model for* f *and* f'*. Here* $t_S = \mathcal{O}(q^2)$*,* $\epsilon = \mathcal{O}((q+\mu)^4/2^n)$ *and the result holds for any* t_D*.*

5.3 VIL-PRF Using the Two-Key Construction

Recall that to prove that the three-key enciphered CBC $H[f_1, f_2, f_3]$ is a good domain extender of PRFs, we reduced its security to the security of encrypted CBC, by simply simulating the invocations of f_2 (which are present in the enciphered, but not in the encrypted CBC mode). This does not work for $H_\alpha[f, f']$, as we can't simulate f because we do not know its key (in the three key case, f_2 and f_1 used independent keys, so this was possible). So one has to do a direct proof. In the full version of this paper we prove the following Theorem (we give a high level sketch of the proof in Section 6.3).

Theorem 6. *Let* $f : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ *be a* $(t, 2\mu, 2\mu n, \epsilon)$*-secure FIL-PRF family. Then* $H_\alpha[f_k, f_{k'}](.)$ *(where* k, k' *is the secret key) is a* $(t', q, \mu n, 4\mu^2/2^n + 2\epsilon)$*-secure VIL-PRF family where* $t' = t - O(\mu n)$*.*

5.4 CRHF Using the Two-Key Construction

In order to prove the collision-resistance of the two-function construction $H_\alpha[f, f']$, we essentially need to show that the XOR compression function $g_\alpha[f]$ is collision-resistant, since it is trivial to find a length-preserving collision-resistant function f' and we use MD strengthening in this construction (similar to Lemma 2). If we make a suitably strong assumption (for instance, f is a FIL-RO), then we can show that $g_\alpha[f]$ is a FIL-RO. We give a proof of this lemma in the full version of this paper.

Lemma 8. *Let* $f : \{0,1\}^n \rightarrow \{0,1\}^n$ *be a length preserving function. The XOR compression function* $g_\alpha[f]$ *is* (t, ϵ)*-secure collision resistant function in the FIL-RO model for* f*. Here* $\epsilon = \mathcal{O}(q^4/2^n)$*, where* q *is the number of FIL-RO queries made by the attacker to* f*.*

6 Single-Key Enciphered CBC Construction

Finally, we show how to further optimize our mode to use only a single length-preserving function f. The first natural idea is to start with the two-key mode from the previous section, and then simply make the second function $f' = f$. It is easy to see that this does not affect the collision-resistance much (since the "outer function" f' did not do anything there anyway). Unfortunately, this change makes our mode insecure. In essence, the reason is due to the fact that our (suffix-free) encoding is not prefix-free, and so called

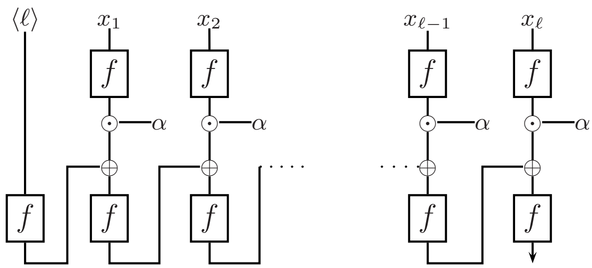


Fig. 4. The single-key enciphered CBC construction $H_\alpha[f]$ for constructing MAC and PRF

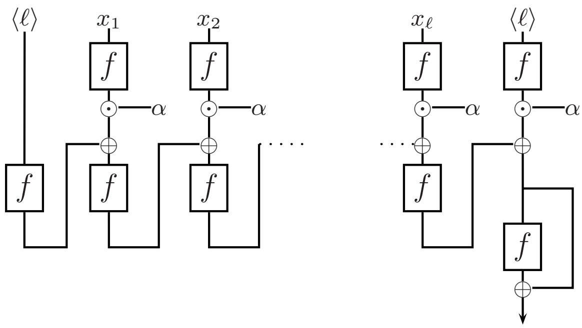


Fig. 5. The "enhanced" single-key enciphered CBC construction $H_\alpha[f]'$ for constructing RO

"extension attacks" become possible. (This is quite analogous to the usual CBC-MAC [3] and cascade constructions [5] which are only secure for prefix-free inputs.)

CONSTRUCTION FOR PRFS AND MACS. Luckily, it turns out that if instead of appending the input length, we prepend it (to get a *prefix-free encoding*) then the resulting construction can be proven secure (with $f' = f$) for the "keyed" setting of MACs and PRFs. The resulting construction, depicted in Figure 4 and denoted $H_\alpha[f]$, is formally defined below:

$$H_\alpha[f](x_1 \parallel \ldots \parallel x_\ell) \stackrel{\text{def}}{=} f\left(G_\alpha[f](\langle \ell \rangle \parallel x_1 \parallel \ldots \parallel x_\ell\right)$$

CONSTRUCTION FOR VIL-RO. Unfortunately, the above construction is still not enough for the question of building a VIL-RO from a single FIL-RO. To handle this case as well, we need to modify the two-key construction as follows:

1. Instead of setting $f' = f$, we use the Davies-Mayers-type construction $f'(x) = f(x) \oplus x$.
2. We still keep the suffix-free encoding (by appending the number of blocks to the input), but now also ensure the prefix-free encoding by prepending the number of blocks to the input.

5), and formally define it on input $X = x_1 \parallel \ldots \parallel x_\ell$ as follows:

$$H_\alpha[f]'(X) \stackrel{\text{def}}{=} f(G_\alpha[f](\langle \ell \rangle \parallel X \parallel \langle \ell \rangle)) \oplus G_\alpha[f](\langle \ell \rangle \parallel X \parallel \langle \ell \rangle)$$

We remark that although this final construction $H_\alpha[f]'$ is defined for building VIL-RO (for which the simpler construction $H_\alpha[f]$ is not enough), it is easy to extend the MAC/PRF security of $H_\alpha[f]$ to show that $H_\alpha[f]'$ also works for the case of MACs and PRFs. For the sake of elegance, though, we only analyze the simpler variant $H_\alpha[f]$ when studying the domain extension of PRFs and MACs.

6.1 Single-Key VIL-MAC Construction

To prove that the one-key enciphered CBC $H_\alpha[f]$ is a good domain extension for MACs, we cannot apply the methodology of An and Bellare (as explained in Section 3.1) that we used for the three and the two key construction. Recall that in this methodology, one first proves that the construction (ignoring the last invocation of f) is weakly collision resistant, and then the final application of f (with an independent key) gives us the MAC property. In $H_\alpha[f]$ there is no final invocation of f with an independent key. Instead, in the full version of the paper, we give a direct reduction to prove the following Theorem.

Theorem 7. *If the function family f is a $(t, 3\mu, 3\mu n, \epsilon)$-secure MAC family, then $H_\alpha[f_k]$, where k is the secret key, is a $(t', q, \mu n, \epsilon \cdot 49 \cdot \mu^4)$-secure MAC where $t' = t - O(\mu n)$ and q is arbitrary.*

6.2 Single-Key VIL-RO Construction

As discussed above, the single-function RO construction $H_\alpha[f]'$ is slightly different from the MAC and PRF case. We show that this construction is indifferentiable from a VIL-RO. The formal proof of this theorem is more involved than the two/three FIL-RO case. In particular, the proof of indifferentiability crucially uses the "extractability" of the Davies-Meyer construction in the end of the "enhanced" enciphered CBC construction. We defer the formal proof to the full version of this paper [13].

Theorem 8. *Consider a length-preserving function $f : \{0,1\}^n \rightarrow \{0,1\}^n$. Then the single-function RO construction $H_\alpha[f]'$ is $(t_D, t_S, q, , \mu, \epsilon)$-indifferentiable from a random oracle in the FIL-RO model for f. Here $t_S = \mathcal{O}(q^2)$, $\epsilon = \mathcal{O}((q+\mu)^4/2^n)$ and the result holds for any t_D.*

6.3 Single-Key VIL-PRF Construction

We prove that our single-key enciphered CBC construction $H_\alpha[f]$ is a secure domain extension for PRFs by adapting the proof for "plain" prefix-free CBC of Maurer (Theorem 6 in [15]). The situation here is somewhat more complicated than in the three and two key cases considered so far. There, security can be proven using the following high level idea: first one proves that the construction (ignoring the final invocation of f) is (computationally) almost universal (see [2]); i.e. any two *fixed* messages are unlikely to collide. And this is enough to prove security because of a final invocation of an independent PRF. For $H_\alpha[f]$, this proof idea does not directly work, as there is no final invocation with an f using an independent key. Fortunately, one can use a powerful theorem (Theorem 2 from [15]) to still argue security in our setting as well. Details are deferred to the full version [13].

Theorem 9. *Let* $f : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ *be a* $(t, 3\mu, 3\mu n, \epsilon)$*-secure FIL-PRF family. Then* $H_\alpha[f_k](.)$ *(where* k *is the secret key) is a* $(t', q, \mu n, 9\mu^2/2^n + 2\epsilon)$*-secure VIL-PRF family where* $t' = t - O(\mu n)$.

Acknowledgments. We would like the thank Dan Boneh, Marc Fischlin and Phillip Rogaway for several very useful conversations in the early stages of this work.

References

1. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
3. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)
4. Bellare, M., Pietrzak, K., Rogaway, P.: Improved Security Analyses for CBC MACs. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 527–545. Springer, Heidelberg (2005)
5. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom Functions Re-visited: The Cascade Construction and Its Concrete Security. In: Proc. 37th FOCS, pp. 514–523. IEEE, Los Alamitos (1996)
6. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
7. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, Springer, Heidelberg (2006)
8. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, Springer, Heidelberg (2002)
9. Black, J., Cochran, M., Shrimpton, T.: On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 526–541. Springer, Heidelberg (2005)
10. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
11. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
12. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, Springer, Heidelberg (2004)
13. Dodis, Y., Pietrzak, K., Puniya, P.: A New Mode of Operation for Block Ciphers and Length-Preserving MACs. Full version of this paper (2008)
14. Dodis, Y., Puniya, P.: l Feistel Networks Made Public, and Applications. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 534–554. Springer, Heidelberg (2007)

15. Maurer, U.: Indistinguishability of Random Systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
16. Maurer, U., Sjodin, J.: Single-key AIL-MACs from any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, Springer, Heidelberg (2005)
17. Maurer, U., Tessaro, S.: Domain Extension of Public Random Functions: Beyond the Birthday Barrier. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 187–204. Springer, Heidelberg (2007)
18. Merkle, R.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
19. Petrank, E., Rackoff, C.: CBC MAC for Real-Time Data Sources. J. Cryptology 13(3), 315–338 (2000)
20. Pietrzak, K.: A Tight Bound for EMAC. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 168–179. Springer, Heidelberg (2006)
21. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, Springer, Heidelberg (1994)
22. Rogaway, P., Steinberger, J.: How to Build a Permutation-Based Hash Function. In: Dagstuhl workshop (September 2007)
23. Rogaway, P., Steinberger, J.: Security/Efficiency Tradeoffs for Permutation-Based Hashing. In: Eurocrypt 2008, Istanbul, Turkey (April 2008)
24. Shrimpton, T., Stam, M.: Building a Collision-Resistant Compression Function from Non-Compressing Primitives, Cryptology ePrint Archive: Report 2007/409
25. Simon, D.R.: Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)

Security/Efficiency Tradeoffs for Permutation-Based Hashing

Phillip Rogaway[1] and John Steinberger[2]

[1] Department of Computer Science, University of California, Davis, USA
[2] Department of Mathematics, University of British Columbia, Canada

Abstract. We provide attacks and analysis that capture a tradeoff, in the ideal-permutation model, between the speed of a permutation-based hash function and its potential security. We show that any $2n$-bit to n-bit compression function will have unacceptable collision resistance it makes fewer than three n-bit permutation invocations, and any $3n$-bit to $2n$-bit compression function will have unacceptable security if it makes fewer than five n-bit permutation invocations. Any rate-α hash function built from n-bit permutations can be broken, in the sense of finding preimages as well as collisions, in about $N^{1-\alpha}$ queries, where $N = 2^n$. Our results provide guidance when trying to design or analyze a permutation-based hash function about the limits of what can possibly be done.

1 Introduction

Overview. Consider the problem of constructing a cryptographic hash function where, for reasons of speed, assurance, or minimalism, you've decided to base your design on an off-the-shelf blockcipher, say AES, with an $n = 128$ bit block-size and a small, fixed set of keys. To keep things modular, you've decided to first build a $3n$-bit to $2n$-bit compression function from your n-bit permutations $\pi_1, \ldots, \pi_k$. You plan to prove your construction sound in the ideal-permutation model, where the adversary has black-box access to the forward and backwards direction for each π_i.

Perhaps surprisingly, the design problem just described is extremely challenging. If you write a construction down, chances are good that, after a while, you'll find an efficient attack. It's quite unlikely you'll find an easy proof. At least this was our experience, and over a period of many months.

In this paper we partially explain [illegible] the design difficulty is coming from. Basically, the problem is that [illegible] In particular, compressing $3n$ bits to $2n$ bits needs at least [illegible] permutation invocations just to break the birthday bound of $N^{0.5}$ queries (where $N = 2^n$) that motivates having a double-length construction in the first place. And even with five permutations there is [illegible] going to be a collision-finding attack that uses about $N^{0.6}$ queries, which isn't all that great.

In prior work, Black, Cochran, and Shrimpton [1] showed that any rate-1 iterated hash function whose compression function uses a single permutation

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 220–236, 2008.

call must be insecure in the ideal-permutation model.[1] In the present work, the Black [illegible] result is seen as a point on a continuum: while one permutation call is not enough, more and more calls buys you, potentially, better and better security. Concretely, we exhibit a quantifiable tradeoff between the number of permutation calls and the effectiveness of a corresponding attack. The attack's effectiveness diminishes rather slowly with the number of permutation calls.

The problem of constructing a cryptographic hash function from a fixed-key blockcipher dates to Preneel, Govaerts, and Vandewalle [8]. They explain the utility of this problem and specify a family of solutions with inverse rates of 4–8. For the concrete parameters they suggest, a compression function mapping 310 bits to 256 bits using four calls to 64-bit permutations, our own pigeonhole-birthday attack (Theorem 2) implies that an adversary will probably have the information it needs to construct a collision after making just two million queries. While this doesn't mean that there's a computationally efficient way to [illegible] the desired collision, it does mean that, for the stated parameters, one can't possibly prove a decent security bound in the random-permutation model.

We want to emphasize at the outset that this paper is about attacks, not constructions or their security proofs. It remains an intriguing open question if, for every choice of parameters, there [illegible] a construction whose provable security matches that given by our attacks. Our guess is that the answer is [illegible], which would mean that the results of this paper are tight.

Our results and their interpretation. Let us now summarize our results one-by-one. First we look at the collision resistance of a permutation-based compression function. We show that if a compression function maps mn bits to rn bits using k calls to n-bit permutations—a signature we abbreviate as $m \xrightarrow{k} r$, eliding n—then an adversary will be able to find a collision using some[2] $N^{1-(m-0.5r)/k}$ queries, where, again and throughout, $N = 2^n$. In particular, a $2 \xrightarrow{2} 1$ compression function can be broken with about $N^{1-(2-0.5)/2} = N^{1/4}$ queries, which is unacceptably few, while a $3 \xrightarrow{4} 2$ compression function can be broken in about about $N^{1-(3-1)/4} = N^{1/2}$ queries, which, for a double-length construction, is again too few.

Our bounds suggest a qualitative difference in behavior between the $m \xrightarrow{k} 1$ (single-length) and the $m \xrightarrow{k} 2$ (double-length) settings: in the first case $k = 3$ permutations is enough to potentially achieve the optimal security of $N^{1/2}$ queries, while in the second case no number of permutation calls can ever achieve the optimal security of N queries. It has recently been shown that one [illegible] asymptotically achieve the optimal security of $N^{1/2}$ queries with a $2 \xrightarrow{3} 1$ compression function [9], one of the rare choices of parameters for which a $m \xrightarrow{k} r$ construction is known to have a security bound matching that of our attacks.

---

[1] The *rate* of a permutation-based hash function is α if it processes αn bits worth of data with each n-bit permutation invocation. The inverse rate $\beta = 1/\alpha$ is therefore the number of permutation calls used per n bits of input.

[2] In summarizing our results we omit distracting multiplicands or addends that have a second-order effect.

Next we put compression functions aside and look at collision resistance for a full-fledged permutation-based hash function $H\colon \{0,1\}^* \to \{0,1\}^{rn}$. We show that if the rate of the hash function is α then an adversary can find collisions with about $N^{1-\alpha}$ queries. In particular, rate-1 hash functions are completely insecure, as already discovered by Black [illegible] for the special case of iterated hash functions using a single permutation call per iteration. In addition, a rate-1/2 double-length hash function ($r = 2$) will admit an $N^{1/2}$-query attack. As this is what one expects from a single-length construction, the conclusion is that a double-length construction must have a rate of less than 1/2.

We also look at the preimage resistance of permutation-based compression functions and hash functions. In the former case, a preimage for an $m \xrightarrow{k} r$ construction can be found in about $N^{1-(m-r)/k}$ queries. In particular, preimages can be found in any $2 \xrightarrow{3} 1$ design with about $N^{2/3}$ queries. (Happily, the $2 \xrightarrow{3} 1$ construction we mentioned asymptotically matches this bound [9].) So while collision-resistance can be "as good as a random function" with a $2 \xrightarrow{3} 1$ design, no such design can be comparably good with respect to preimage resistance. For a full-fledged rate-α hash function, a preimage can be found in about $N^{1-\alpha}$ queries, which is, rather oddly, the same as for collision resistance.

In a somewhat different spirit, Section 8 of this paper considers the number of bits that a permutation-based compression function must keep in memory in order to be collision resistant. We show that an $m \to r$ compression function must, at some point during its computation, keep strictly more than mn bits in memory, or else it will suffer from devastating attacks. If we imagine that the compression function is built from n-bit [illegible] connecting the permutations, then the compression function must, at some point, maintain at least $m+1$ active wires to have any hope for collision resistance.

Appendix A sketches a generalization of the attack of Black [illegible] Theirs is a collision attack on permutation-based iterated hash functions that use a single permutation call per iteration; here we adapt it to the case where k permutation calls are made per iteration. The attack is only applicable to iterated hash functions, and our version of it uses a heuristic assumption, but the bound is slightly better than that of our attack for an arbitrary hash function.

2 The Model

Consider a compression function $H\colon \{0,1\}^{mn} \to \{0,1\}^{rn}$ built from black-box n-bit permutations, where $m > r \geq 1$ and $n \geq 1$. Let us assume that for H to process its mn-bit input requires making k calls, in order, to permutations $\pi_1, \ldots, \pi_k\colon \{0,1\}^n \to \{0,1\}^n$. Then H necessarily takes the form illustrated in Fig. 1, for some sequence of functions $f_1, \ldots, f_k, g$. Along with permutations $\pi_1, \ldots, \pi_k\colon \{0,1\}^n \to \{0,1\}^n$, functions $f_i\colon \{0,1\}^{imn} \to \{0,1\}^n$ ($i \in [1..k]$) and $g\colon \{0,1\}^{(i+1)mn} \to \{0,1\}^{rn}$ define H. In general, we do not require anything of $f_1, \ldots, f_k, g$ beyond their having the specified domain and range.

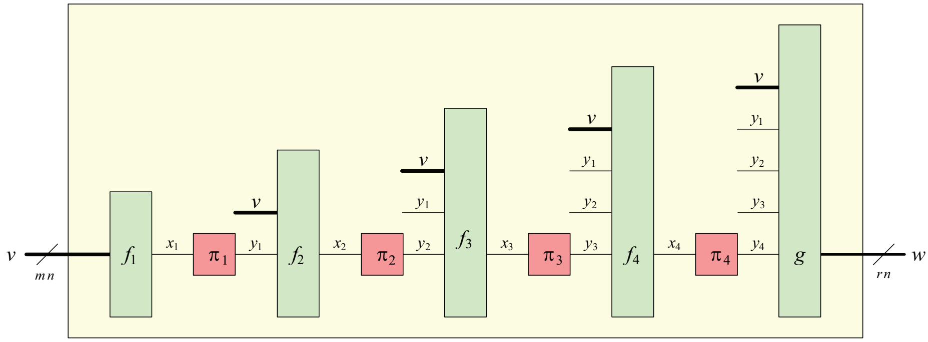
Because $\pi_1, \ldots, \pi_k$ are always called in the order π_1 and then π_2 and so forth, up to π_k, we call the model just described the [illegible] model. It includes designs where the permutations $\pi_1, \ldots, \pi_k$ are unrelated—the [illegible] [illegible] setting—and designs where a single permutation π $(= \pi_1 = \cdots = \pi_k)$ is always called—the [illegible] setting. It does not include the case where the identity of the permutation (ie, which π_i is used at each step) is data dependent. This restriction turns out not to be so significant—more on that in just a bit.

Let H be a fixed-order compression function, notation as above, and let $\mathcal{A}$ be an adversary with access to oracles $\pi_1, \ldots, \pi_k$ (and, in principle, their inverses—only that this isn't needed in any of our attacks). The [illegible] of $\mathcal{A}$ in finding collisions in H is the probability that $\mathcal{A}$ asks a sequence of queries such that there exist distinct inputs $v, v' \in \{0,1\}^{mn}$ for which the adversary has asked all necessary queries to compute $H(v)$ and $H(v')$. This probability is over the adversary's coins and over uniform permutation oracles $\pi_1, \ldots, \pi_k$. (This sentence assumes the distinct-permutation setting. More generally, select a single random permutation to model each distinct π_i.) Note that we do not insist that the adversary actually output a collision: we assert that it wins if a computationally-unbounded adversary [illegible] compute a collision from what it knows. It is true that this makes the attacks less "realistic" than if we had paid attention to the attacker's time and required it to print out its collision. But since our main goal is to understand the limits of what is provably secure in the random-permutation model, we can ignore time and adopt a liberal notion of adversarial success.

As mentioned already, one can generalize the fixed-order model by letting the compression function choose which permutation to invoke at each step: in Fig. 1, add in a line 3.5 saying $j \leftarrow e_i(v, y_1, \ldots, y_{i-1})$, and use j, not i, as the subscript for π at line 4. This [illegible] model was employed by Black, Cochran, and Shrimpton [1]. We ourselves prefer the fixed-order model, and assume it for quantitative results. Philosophically, letting permutation selection vary according to the data being hashed would make permutation-based hashing conceptually coincide with blockcipher-based hashing, contrary to the point of our investigation. More pragmatically, good lower bounds in the (simpler) fixed-order setting are already enough to imply good lower bounds in the (more complex) no-fixed-order setting. To see this, note that if H is a no-fixed-order compression function that makes k permutation calls, then there's a functionally identical fixed-order compression function H' that makes k^2 calls: H' just queries its k permutations in a round-robin fashion. Because of this, lower-bounds applicable to (the fixed-order) H' are inherited by (the no-fixed-order) H if one simply replaces each k by k^2. Since we are always thinking of k as a small constant, the quantitative change in bounds is not so significant. In particular, every qualitative conclusion that we draw in this paper is an accurate interpretation of our results for the fixed-order model and the no-fixed-order model, too.

3 The Trivial Attacks

We begin by acknowledging two trivial but nonetheless significant attacks on any permutation-based compression function, the [illegible] and the [illegible]




```
algorithm H(v)
for i ← 1 to k do
    x_i ← f_i(v, y_1, ..., y_{i-1})
    y_i ← π_i(x_i)
w ← g(v, y_1, ..., y_k)
return w
```

Fig. 1. Illustration and definition for a permutation-based compression function. Regarding $\pi_1, \ldots, \pi_k$ as oracles, functions $f_1, \ldots, f_k$ and g define the scheme, which maps an mn-bit input v to an rn-bit output w.

The former attack asks all kN possible queries, where $N = 2^n$. At that point the hash of message will be known and so, by the pigeonhole principle (remember that $m > r$), there will be messages known to collide. This implies that it is, in some sense, futile to select an output length exceeding $2n$ bits, as $2n$ bits are already enough to accommodate the maximum feasible security[3]. With an output length of $3n$ bits, for example, you'll never get a construction withstanding anything near the optimal value of $q = N^{3/2}$ queries, as no construction can withstand more than $q = N^{1+(\lg k)/n} \ll N^{3/2}$ queries (the "$\ll$" is because we assume that k is a small number).

The is to compute the permutations necessary to hash $p = q/k$ random messages. By the birthday phenomenon, one expects to see a collision when $p \approx \sqrt{2 \ln 2}\, N^{r/2} \approx 1.18\, N^{r/2}$. For a proper upperbound, note that when $N \geq 2^{16}$, which we will henceforth implicitly assume, the probability of a collision is at least $1/2$ if $p \geq 1.18\, N^{1/2}$ balls are randomly and uniformly thrown into N bins. We record the efficacy of our two attacks in the following proposition.

Proposition 1. $H\colon \{0,1\}^{mn} \to \{0,1\}^{rn}$ k $N = 2^n$

$q = kN$ 1

$q = 1.18kN^{r/2}$ $\geq 1/2$

[3] This is assuming an information-theoretic adversary, whose only cost is the number of queries made; a "real adversary" may well be hindered by a longer output.

In all theorem statements where, like above, q is an integer but the quantity on the right may be fractional, it is implicit that q is obtained by rounding up the expression on the right. Also, here and subsequently, it is not necessary to restrict m and r to natural number; it is fine to select any rational values m and r such mn and rn are positive integers.

4 The Pigeonhole Attack

We now give a more interesting collision attack on compression functions. It succeeds, always, in about $kN^{1-(m-r)/k}$ queries.

Theorem 1. $H\colon \{0,1\}^{mn} \to \{0,1\}^{rn}$ k $N = 2^n$

$q = k\,(N^{1-(m-r)/k} + 1) \approx k\,N^{1-(m-r)/k}$

H 1 □

The concrete consequences of this are interesting. Suppose H is a $2 \xrightarrow{1} 1$ compression function. Then it can be broken in just $q = 2$ queries. So $k = 1$ permutation calls certainly won't do, as shown by Black, Cochran, and Shrimpton [1] in the iterated hash-function setting. In addition, we see that a $2 \xrightarrow{2} 1$ compression function can be broken in about $N^{1/2}$ queries, which is optimal for a hash function of output length n, except that Theorem 1 states the collision can be found with probability 1, whereas an ideal construction would require $2N$ queries for the same result. Quantitative results are tabulated in the top half of Fig. 2.

Let $p = \lfloor q/k \rfloor$. In brief, the adversary chooses p queries to make to π_1 that enable him to "start" hashing the largest possible number of inputs (each input requires a π_1 query); then the adversary chooses p queries to make to π_2 that will enable him to continue hashing the largest possible number of inputs up to and including the π_2 step; and so on for π_3, ..., π_k. If, at the end, the adversary is still able to hash more than N^r inputs, then the adversary wins because some two inputs necessarily collide. The proof simply consists of computing how large p must be for the latter event to happen.

Note first the observation that if B balls are thrown into N bins the $p \le N$ most occupied bins must contain at least pB/N balls. We will repeatedly use this observation below. Now with the hash function H specified by $f_1, \ldots, f_k$, g, choose a p-element set $X_1 \subseteq \{0,1\}^n$ that has a maximum number of preimages under f_1. By the observation just made, this maximum number of preimages is at least $pN^m/N = pN^{m-1}$ points. The adversary will ask for π_1 at each point $x_1 \in X_1$. The adversary has so far made p queries and there are at least pN^{m-1} points $v \in \{0,1\}^{mn}$ for which the adversary knows how to compute the first permutation in the hash chain. Call this set of points V_1. So $|V_1| \ge pN^{m-1}$ and for each point $v \in V_1$ the adversary knows the corresponding x_1, y_1, and x_2. Next choose p points $X_2 \subseteq \{0,1\}^n$ with a maximum number of $v \in V_1$ that give rise to an $x_2 \in X_2$. Again by the observation that began this paragraph, this set of points V_2 has cardinality $|V_2| \ge p|V_1|/N \ge p^2N^{m-2}$. Continue in this way, selecting a set V_3 where $|V_3| \geq p^3 N^{m-3}$ and making p more queries so that the adversary will know how to compute the beginning computations of a hash value for everything in V_3, knowing everything up to and including the third permutation π_3. Continue until the adversary constructs a set V_k where $|V_k| \geq p^k N^{m-k}$ and the adversary knows how to hash everything in V_k all the way until the end.

If $|V_k| \geq p^k N^{m-k}$ exceeds N^r then, by the pigeonhole principle, there must be two values in V_k that have the same hash, and this hash is known by the adversary we have constructed. Thus the adversary will succeed in finding a collision if $p^k > N^{r-m+k}$, which is to say that it necessarily succeeds if $p > N^{(r-m+k)/k} = N^{1-(m-r)/k}$. So the adversary will find a collision if $\lfloor q/k \rfloor$ exceeds $N^{1-(m-r)/k}$ (hence the chosen value of q). This completes the proof. ∎

5 The Pigeonhole-Birthday Attack

In the proof above we used the fact that a collision is guaranteed as soon as $|V_k| \geq p^k N^{m-k} > N^r$. But it seems unlikely that one would really have to wait so long as that; if the H-outputs computed by the adversary had been random then, by the birthday phenomenon, one would expect to see a collision around the time that $|V_k| = N^{r/2}$, or to be quite exact around the time that $|V_k| = 1.18 N^{r/2}$. Let us [illegible] that the hash function outputs computed by the adversary in the proof of Theorem 1 behave no worse than random outputs with respect to the appearance of collisions. Call this the [illegible]. Then solving for the integer p in $p^k N^{m-k} \geq 1.18 N^{r/2}$ reveals that we expect to see a collision after $q = kp = k\lceil (1.18)^{1/k} N^{1-(m-0.5r)/k} \rceil \leq k(1 + (1.18)^{1/k} N^{1-(m-0.5r)/k}) \leq k(1 + 1.18\, N^{1-(m-0.5r)/k}) \approx kN^{1-(m-0.5r)/k}$ queries, an improvement from the earlier $q \approx kN^{1-(m-r)/k}$ by a multiplicative factor of $N^{r/2k}$. To summarize:

Theorem 2. [illegible] $H\colon \{0,1\}^{mn} \to \{0,1\}^{rn}$ [illegible] k [illegible] $N = 2^n$ [illegible]

$$q = k(1 + (1.18)^{1/k} N^{1-(m-0.5r)/k}) \approx k\, N^{1-(m-0.5r)/k}$$

[illegible] $\geq 1/2$ □

The stated bound suffers from a peculiar behavior in the $2 \xrightarrow{k} 1$ case when $k \geq 4$, whence the theorem states that $q \approx kN^{1-3/2k} \geq kN^{5/8}$ queries are sufficient for the attack described, but Proposition 1 ensures that $q = 1.18\, kN^{1/2}$ queries was already enough. The gap may seem more puzzling considering that the pigeonhole-birthday attack [illegible] a type of birthday attack and, under the uniformity assumption, it cannot do worse than what Proposition 1 guarantees. The problem can be traced to the $p^k N^{m-k}$ lower bound for the number of outputs obtained by the pigeonhole attack, which, in turn, stems from the observation made at the beginning of Theorem 1 that when B balls are thrown into N bins, the $p \leq N$ most occupied bins must contain at least pB/N balls. In fact one can strengthen this observation by noting that the $p \leq N$ most occupied bins must contain at least $\mu_{p,N}(B)$ balls, where $\mu_{p,N}(B)$ is $p\lceil B/N \rceil$ if $p \leq B \bmod n$ or $B \equiv 0 \bmod n$,

atk	adv	asmp	$m \to r$	$\approx$ bound	1	2	3	4	5	6	8
ph	1	no	$2 \to 1$	$kN^{1-1/k}$	2	$2^{65.0}$	$2^{86.9}$	$2^{98.0}$	$2^{104.7}$	$2^{109.3}$	2^{115}
ph	1	no	$3 \to 2$	$kN^{1-1/k}$	2	$2^{65.0}$	$2^{86.9}$	$2^{98.0}$	$2^{104.7}$	$2^{109.3}$	2^{115}
pb	0.5	yes	$2 \to 1$	$1.18kN^{1-3/2k}$	2	$2^{33.1}$	$2^{65.7}$	$2^{66.2}$	$2^{66.6}$	$2^{66.8}$	$2^{67.2}$
pb	0.5	yes	$3 \to 2$	$1.18\,kN^{1-2/k}$	1	3	$2^{44.3}$	$2^{66.1}$	$2^{79.2}$	$2^{88.0}$	$2^{99.0}$

Fig. 2. Attacks on an $m \xrightarrow{k} r$ compression function. Columns are the attack (ph for pigeonhole, pd for pigeonhole-birthday), the adversary's advantage, whether a heuristic assumption is used in the analysis, the compression parameters, the approximate value of q to get this advantage, and numerical values for various values of k, all with $n = 128$.

and $p\lfloor B/N \rfloor + B \bmod N$ otherwise. One thus gets at least $\mu^{(k)}_{p,N}(N^m)$ outputs from the pigeonhole attack (the k-th iterate of the function), better than the approximation $p^k N^{m-k}$. To find the "real" p needed by the attack one can solve for the least integer p such that $\mu^{(k)}_{p,N}(N^m) \geq 1.18N^{r/2}$. As this is somewhat hard to compute, an alternative is to note that, at the end of the pigeonhole-birthday attack, there are at least $p = \lfloor q/k \rfloor$ strings that the adversary knows how to hash, and so $p = 1.18N^{r/2}$ queries are enough (still under the uniformity assumption). We can therefore sharpen the statement of Theorem 2 to select q as the minimum of the current value of q and $1.18\,kN^{r/2} + k \approx 1.18\,kN^{r/2}$, since $p = \lfloor q/k \rfloor > q/k - k$. In Fig. 2 we use this tighter bound to compute the third-row entries.

Interpretation. The bound of the pigeonhole-birthday attack is illustrated numerically in Fig. 2 for $n = 128$ bits. For $2 \to 1$ hashing the analysis indicates that, with $k = 2$ permutations, a collision will be found in around $N^{1/4}$ queries. This is excessively low, making $k = 3$ permutations the best one can hope for in this case. With $k = 3$ permutations the bound jumps to around $N^{1/2}$ queries, which is of course optimal for a hash function producing an n-bit output. This suddenly-optimal behavior is qualitatively different from what happens when the output length is $2n$ bits or more, in which case more permutation calls (potentially) buys more security, but where optimal collision resistance can never reached. For $3 \to 2$ hashing the adversary can break the construction in around $q = N^{1-2/k}$ queries. Since a double-length construction ought to withstand significantly more than $N^{1/2}$ queries (otherwise, it makes more sense to use a single-length construction), the conclusion is that $k = 5$ permutations is the minimum number of calls that makes sense for $3 \to 2$ hashing.

It should be noted that, because of the uniformity assumption, the analysis of Theorem 2 is essentially heuristic. But assumptions analogous to the uniformity assumption are routinely made when analyzing cryptographic attacks, sometimes without even mention that an assumption is being made. And of course one expects that a good hash function [illegible] have outputs that look uniform on any natural set of inputs produced by an attack.

6 Attacks on Rate- Constructions

Theorems 1 and 2 can be recast in terms of what they say about a permutation-based hash function with a given rate (as opposed to what they say about a compression function with a given number of blockcipher calls). Let $H : \{0,1\}^* \to \{0,1\}^{rn}$ be a fixed-order hash function based on an n-bit permutation. This means that the algorithm is of the form specified in Fig. 1, except that the input message v may now have any length, and sequences $\pi_1, \pi_2, \pi_3, \ldots$ and $f_1, f_2, f_3, \ldots$ are thought of as infinite, and the number k of permutation invocations is a function $k = k(v)$ of the input v. Then we say that H has α if α is the largest real number such that hashing a message M requires at most $|M|/\alpha n$ permutation calls. (One could also add in an additive constant δ to account for padding or other extra work done at the end of processing the message.) The , $\beta = 1/\alpha$, is the number of permutation calls per n-bits of message processed; hashing M requires at most $\beta|M|/n$ permutation invocations. We now show that the pigeonhole and pigeonhole-birthday attacks imply a tradeoff between the (potential) security of a permutation-based hash function and its rate.

Theorem 3. H: $\{0,1\}^* \to \{0,1\}^{rn}$ $\alpha = 1/\beta$ $N = 2^n$

$$q = \lfloor \beta \lceil \ln(2)\alpha n r + \alpha \rceil \rfloor (eN^{1-\alpha} + 1) \approx 1.89\, nrN^{1-\alpha}$$

1 □

For any $m \geq 1$ we can restrict H to inputs of length mn, whence H becomes a compression function H': $\{0,1\}^{mn} \to \{0,1\}^{rn}$ that makes at most $k = \lfloor \beta m \rfloor$ permutation calls. By Theorem 1, a collision for this compression function can be found with probability 1 in $k(N^{1-(m-r)/k}+1) \leq k(N^{1-\alpha+r/k}+1)$ queries, where again $k = \lfloor \beta m \rfloor$ (the inequality holds because $\alpha \leq m/k$). We set $m = \lceil \ln(2)\alpha n r + \alpha \rceil$ so $k = \lfloor \beta \lceil \ln(2)\alpha n r + \alpha \rceil \rfloor$ (chosen by calculus to minimize $kN^{1-\alpha+r/k}$). Then $k \geq \beta\lceil \ln(2)\alpha n r + \alpha\rceil - 1 \geq \beta(\ln(2)\alpha n r + \alpha) - 1 = \ln(2) n r$ and $N^{r/k} \leq N^{1/\ln(2)n} = e$, so $k(N^{1-\alpha+r/k}+1) \leq \lfloor \beta \lceil \ln(2)\alpha n r + \alpha \rceil \rfloor (eN^{1-\alpha}+1)$, as desired. ∎

One can improve the constant of 1.89 in Theorem 3 by employing the bound of Theorem 2 instead of Theorem 1. Then choosing $m = \lceil ((\ln 2)/2)\, \alpha n r + \alpha \rceil$ yields a final (approximate) bound of $0.94\, nrN^{1-\alpha}$ queries (for generating a collision with probability at least 1/2). Besides halving the probability of success, the price of this change is that one would now need to make the uniformity assumption on the hash function, inherited from Theorem 2, for mn-bit strings.

Ignoring the leading multiplicative and additive factors in Theorem 3 we can summarize the result as saying that any rate-α permutation-based hash function will fail when the number of queries gets to around $q = N^{1-\alpha}$. In Fig. 3 we tabulate this more precisely, indicating the sufficient number of queries to break permutation-based hash functions of various rates.

We comment that, in our result, the number of distinct permutations used by the hash function does not matter, as long as they are consulted in a fixed order.

atk	adv	asmp	bound	restrictions	2	3	4	5	6	8
ph	1	no	$1.89\, nr\, N^{1-\alpha}$	none	$N^{0.57}$	$N^{0.74}$	$N^{0.82}$	$N^{0.87}$	$N^{0.90}$	$N^{0.95}$
pb	0.5	yes	$0.94\, nr\, N^{1-\alpha}$	none	$N^{0.56}$	$N^{0.73}$	$N^{0.81}$	$N^{0.86}$	$N^{0.90}$	$N^{0.95}$
tree	0.5	yes	$2\,\beta\, N^{1-\alpha}$	iterated	$N^{0.52}$	$N^{0.69}$	$N^{0.77}$	$N^{0.83}$	$N^{0.86}$	$N^{0.91}$

Fig. 3. Collision-finding attacks on a permutation-based hash function $H\colon \{0,1\}^* \to \{0,1\}^{rn}$ with rate α. The rows are: the attack (pigeonhole, pigeonhole-birthday, tree); the adversary's advantage; whether a heuristic assumption is used in the analysis; the approximate bound; restrictions on the result; and threshold values q when $n = 128$, $r = 2$, and inverse rates $\beta = 1/\alpha \in \{2, 3, 4, 5, 6, 8\}$.

Potentially, the hash function might never reuse the same permutation twice, but it would still suffer from the same vulnerabilities as long as it consulted its permutations in a prescribed order.

7 Attacking Preimage Resistance

We adopt as a notion of preimage resistance that the adversary is presented a random range point $w \in \{0,1\}^{rn}$ and succeeds if it finds (or simply knows from its query history) a preimage to this point. We first observe that our earlier pigeonhole-attack can be adapted so as to become a preimage-finding attack. We then extend this to give an attack on an arbitrary hash function. As we have chosen our range point at random, neither case requires a heuristic assumption.

Theorem 4. $H\colon \{0,1\}^{mn} \to \{0,1\}^{rn}$ k $N = 2^n$

$$q = k\,(N^{1-(m-r)/k} + 1) \approx k\, N^{1-(m-r)/k}$$

$\geq 1/2$ □

The attack proceeds as with the pigeonhole attack, Theorem 1, by greedily constructing a set $V_k \subseteq \{0,1\}^{mn}$ of cardinality at least $p^k N^{m-k}$ for which the adversary knows how to hash everything in V_k. When this set grows to half the size of $\{0,1\}^{rn}$ the adversary will have a 50% chance of inverting a randomly selected point w. So the needed number of queries is the smallest q such that $p^k N^{m-k} \geq 0.5\, N^r$, where $p = \lfloor q/k \rfloor$. Solving, we must ensure that $\lfloor q/k \rfloor \geq (q-k)/k \geq 0.5^{1/k}\, N^{1-(m-r)/k}$. But $0.5^{1/k}\, N^{1-(m-r)/k} \leq N^{1-(m-r)/k}$ so it suffices that $(q-k)/k \geq N^{1-(m-r)/k}$, and the bound follows. ∎

For arbitrary hash functions, as opposed to compression functions, we get the following result to relate preimage resistance to rate.

Theorem 5. $H\colon \{0,1\}^* \to \{0,1\}^{rn}$ $\alpha = 1/\beta$ $N = 2^n$

$$q = \lfloor \beta \lceil \ln(2)\alpha nr + \alpha \rceil \rfloor (eN^{1-\alpha} + 1) \approx 1.89\, nrN^{1-\alpha}$$

$1/2$ □

Proof. The proof is exactly the same as for Theorem 3 since the bounds of Theorem 1 and Theorem 4 are the same. ∎

It is interesting that breaking the preimage resistance of a permutation-based hash function is essentially no harder than breaking its collision resistance; our attacks differ in effectiveness only by a factor of 4. In addition, while one may hope to get near-optimal collision resistance with a $2 \xrightarrow{3} 1$ compression function, the preimage resistance will be nowhere near optimal: preimage-resistance will fail by around $N^{2/3}$ queries, whereas one might hope for something that works up to around N queries. But, as with the collision-resistance of double-length constructions, one can hope to push up the preimage resistance to close to N queries by using more and more permutation calls.

8 The Too-Few-Wires Attack

In this section we switch from considering the number of permutations used by a compression function to considering the amount of memory it requires. Mainly we show that a compression function that maps mn bits to rn bits must keep more than mn bits of information in memory at some point during its computation—otherwise it will offer essentially no collision resistance.

Instead of thinking about memory it is useful to think in terms of *wires*. If we imagine that the compression function is built from n-bit wires connecting the permutations and processed at different points by arbitrary functions, our result implies that at least $m + 1$ wires must be used at some point during the computation—one one more wire than there are input wires.

Naturally one needs to define what it means for a compression function to "keep mn bits in memory" during a computation. The model is as follows: we imagine the mn bits to be kept in m "buckets" of n bits each. At any stage, the buckets may either be processed by an arbitrary function $f_i : \{0,1\}^{mn} \to \{0,1\}^{mn}$; or else one of the buckets may be hit with a permutation π_i, replacing the contents of that bucket with the output of the permutation. The buckets are initialized with the input to the compression function, and the computation is terminated by an arbitrary function mapping $\{0,1\}^{mn}$ to $\{0,1\}^{rn}$.

One may assume that no two functions f_i and f_j are ever applied one right after the other (else one could replace them with their composition), and one can assume that permutations are always applied to the first bucket (as the f_i functions can be used to switch bucket contents). Thus if the compression function uses k permutations $(\pi_1, \ldots, \pi_k)$ and we denote by $\bar{\pi}_i$ the map from $\{0,1\}^{mn}$ to $\{0,1\}^{mn}$ that is π_i on the the first bucket and the identity on all others, then the hash of $v \in \{0,1\}^{mn}$ is $f_k(\bar{\pi}_k(f_{k-1}(\bar{\pi}_{k-1}(\ldots f_0(v)\ldots))))$ where f_k: $\{0,1\}^{mn} \to \{0,1\}^{rn}$ and $f_i : \{0,1\}^{mn} \to \{0,1\}^{mn}$ for $i < k$. Figure 4 shows the basic structure, with buckets drawn as wires. The sequence of permutations $(\pi_1, \ldots, \pi_k)$ may be distinct or include repetitions, but we assume that the

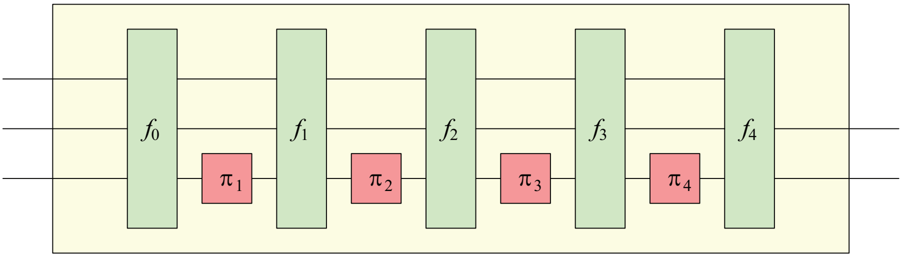


Fig. 4. The structure of a compression function that maps mn bits to rn bits using mn bits of memory where $m = 3$, $r = 2$, and $k = 4$. Each wire represents n bits. Functions f_0, f_1, f_2, f_3, and f_4 are all arbitrary.

permutations are applied in a fixed order, namely that which permutation is applied at a given point does not depend on the contents of the buckets at that point (this restriction can in fact be removed with only a slight increase in the complexity of the attack, so this assumption is mainly made for simplicity). We then have the following:

Theorem 6. [illegible] $H\colon \{0,1\}^{mn} \to \{0,1\}^{rn}$ [illegible] k [illegible] mn [illegible] $2k$ [illegible] □

[illegible] With notation as in the paragraph before Theorem 6, let j be the least number such that f_j is not a permutation. Note that j is well-defined since f_k is not a permutation. Fix any two distinct inputs u and v in $\{0,1\}^{mn}$ such that $f_j(u) = f_j(v)$. Because $f_0, \ldots, f_{j-1}$ are permutations we can compute $u' = f_0^{-1}(\bar{\pi}_1^{-1}(f_1^{-1}(\ldots\bar{\pi}_j^{-1}(u)\ldots)))$ and $v' = f_0^{-1}(\bar{\pi}_1^{-1}(f_1^{-1}(\ldots\bar{\pi}_j^{-1}(v)\ldots)))$ with $2j \leq 2k$ permutation calls. Observe that $f_k(\bar{\pi}_k(f_{k-1}(\bar{\pi}_{k-1}(\ldots f_0(u')\ldots))))$ $= f_k(\bar{\pi}_k(f_{k-1}(\bar{\pi}_{k-1}(\ldots f_0(v')\ldots))))$ since $f_j(u) = f_j(v)$ and we are done. ∎

One can generalize this result. Assume that we have at our disposal k ideal primitives $\rho_1, \ldots, \rho_k$, which are functions from $\{0,1\}^{mn}$ to $\{0,1\}^{mn}$ and such that (i) finding a collision for ρ_i costs q_i expected queries to ρ_i, unless ρ_i is a permutation, in which case (ii) finding a preimage for ρ_i costs one query. (An n-bit permutation can be seen as such a primitive, acting only on the first n bits.) A compression function using (ordered) calls $\rho_1, \ldots, \rho_k$ and mn bits of memory can be modeled as above, with mn-bit to mn-bit functions $f_0, \ldots, f_k$ interwoven with $\rho_1, \ldots, \rho_k$. Then one can easily adapt the proof of Theorem 6 to show that the cost of finding a collision for the compression function is at most $\max(q_i) + 2k$, where the max is taken over all i such that ρ_i is not a permutation, and is defined as 0 if all the ρ_i's are permutations. (Proof: take the least j such that either f_j or ρ_j is not a permutation; in the former case let u, v be colliding inputs of f_j, in the latter case let u, v be colliding inputs of ρ_j paid for with q_j

queries; then push back u, v to inputs u', v' for the original function using the fact that all ρ_i's and f_i's for $i < j$ are permutations.)

This observation has some interesting consequences. For example, say that $\rho_1, \ldots, \rho_k$ are random functions from n bits to n bits, so that it costs $2^{n/2}$ queries to find a collision for given ρ_i. Then a compression function from mn bits to rn bits using mn bits of memory, $m > r$, will have collision resistance of at most $2k + 2^{n/2}$, where k is the number of times the random function is called. This is unsatisfactory if $r \geq 2$. It does not matter whether the random functions are distinct or not, nor how many of them are used.

One can also apply the argument to a blockcipher-based construction, say one with n-bit keys and blocks. First define what it means for a blockcipher to "act" on mn bits: one could assume, say, that the first bucket of n bits is used for the blockcipher's key, that the second bucket of n bits is used for the blockcipher's input, and that the blockcipher's output replaces either the first or second bucket. If the blockcipher's output replaces the key, then the blockcipher application is not a permutation and has collision resistance of $2^{n/2}$ (a collision can be obtained by keeping the word constant and tweaking the key); otherwise the blockcipher application constitutes a permutation. Thus, any mn-bit to rn-bit blockcipher-based compression function using only mn-bits of memory in the sense described has collision resistance of $\sim 2^{n/2}$, which is once again unsatisfactory if $r \geq 2$.

As an example of the findings in this section in action, suppose that someone proposes a $3n$-bit to $2n$-bit compression function as shown in Fig. 4, but where we have 10 rounds and each f_i has some combinatorially strong mixing properties. It will not matter that there are a large number of rounds or that the mixing is strong; the scheme will be breakable in a handful of queries. The issue is that the first collision in any of the f_i's can be "pushed back" through the permutations to make two colliding inputs. Then suppose that, to prevent the pushing back, the designer replaces each $x \mapsto \pi_i(x)$ by the feed-forward gadget $x \mapsto x \oplus \pi_i(x)$. Then the number of required wires has gone up by 1, and the attack is blocked. However if we treat the gadget $x \oplus \pi_i(x)$ as a primitive, the number of wires is back down to 3 and the generalized attack shows that a collision can be found in $2^{n/2}$ queries, or the number of queries necessary to find a collision for the gadget $x \oplus \pi_i(x)$. This is insufficient in a scheme that outputs $2n$ bits.

Finally, we comment that it was not important for the attacks of this section that the input length and output length of the compression be multiples of n; all that matters is that the input has at least one more bit than the output.

Acknowledgments

Most of the work on this paper was carried out while the second author was in the Department of Mathematics at UC Davis. Both authors received funding from NSF grant CCR-0208842 and a gift from Intel; many thanks to Intel, particularly Jesse Walker, and to the NSF, for their kind support.

References

1. Black, J., Cochran, M., Shrimpton, T.: On the impossibility of highly-efficient blockcipher-based hash functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 526–541. Springer, Heidelberg (2005)
2. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
3. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 210–224. Springer, Heidelberg (1993)
4. Hirose, S.: How to construct double-block-length hash functions. The second cryptographic hash workshop (sponsored by NIST) (2006)
5. Knudsen, L., Lai, X., Preneel, B.: Attacks on fast double block length hash functions. Journal of Cryptology 11(1), 59–72 (1998)
6. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
7. Nandi, M.: Towards optimal double-length hash functions. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 77–89. Springer, Heidelberg (2005)
8. Preneel, B., Govaerts, R., Vandewalle, J.: On the power of memory in the design of collision resistant hash functions. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 2002. LNCS, vol. 718, pp. 105–121. Springer, Heidelberg (1993)
9. Rogaway, P., Steinberger, J.: How to build a permutation-based hash function. Manuscript, available from either author's homepage (2008)
10. Satoh, T., Haga, M., Kurosawa, K.: Towards secure and fast hash functions. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences E82-A(1), 55–62 (1999)
11. Shannon, C.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)

A The Tree Attack

This collision-finding attack is applicable only to an iterated hash function. For that setting and with typical parameters, it does a bit better than the pigeonhole-birthday attack. We describe the attack both for that reason and because it generalizes the interesting attack of Black, Cochran, and Shrimpton [1].

When we say that H is an [illegible] permutation-based hash function we mean that it processes one $sn = (m{-}r)n$-bit word of message with each iteration, using a compression function H': $\{0,1\}^{mn} \to \{0,1\}^{rn}$. Hash function H is defined by $H(w_1 \cdots w_\ell) = h_\ell$ where $h_i = H'(h_{i-1} \,\|\, w_i)$ and $h_0 \in \{0,1\}^{rn}$, the [illegible] [illegible], is a constant. The compression function $H'(h, w)$ is $g(h, w, y_1, \ldots, y_k)$ where $x_i = f_i(h, w, y_1, y_2, \ldots, y_{i-1})$ and $y_i = \pi_i(x_i)$. The construction uses k calls to process sn bits, so its rate is $\alpha = s/k = (m-r)/k$. Natural variants to this model, like letting the compression function H' depend on the position index i, are immaterial in the sequel.

As the name suggests, the tree attack is associated to a certain tree, which we will call the [illegible]. The known-hash tree is constructed deterministically from a set of queries. Before describing anything else, we show how to construct the known-hash tree from a set of queries

The known-hash tree is a subtree of an infinite rooted tree called the [illegible]. The full tree has $k+1$ types of nodes, which we denote type 0, type 1, …, type k. A node of type i has children only of type $i+1$, except for a node of type k, which has children of type 0. The root of the full tree has type 0. Nodes of type $1,\ldots,k$ have outdegree N and nodes of type 0 have outdegree N^s. (As usual, $N = 2^n$.) The outgoing edges from nodes of type $1,\ldots,k$ are labeled with all the values from 0 to $N-1$, whereas the outgoing edges from nodes of type 0 are labeled with all the values from 0 to $N^s - 1$. Every node of type 0 has an associated [illegible] in $\{0,1\}^{rn}$ defined inductively as follows: the root has value h_0 and a non-root node v of type 0 has value $g(h, w, y_1, \ldots, y_k)$ where h is the value of the first node u of type 0 on the path from v to the root, and where $w, y_1, \ldots, y_k$ are the values on the edges of the path from u to v. Nodes of type $1,\ldots,k$ also have values, defined as follows: the value of a node v of type $i \geq 1$ is $x_i = f_i(h, w, y_1, y_2, \ldots, y_{i-1})$ where h is the value of the first node u of type 0 on the path from v to the root, and where $w, y_1, \ldots, y_{i-1}$ are the values of the edges on the path from u to v.

This completes the description of the full tree. The known-hash tree is a subtree of the full tree. It is defined from a set of queries $\mathcal{Q} = \{(i_1, x_{i_1}, y_{i_1}), \ldots, (i_q, x_{i_q}, y_{i_q})\}$ made by the adversary, where $\pi_{i_j}(x_{i_j}) = y_{i_j}$ for all $1 \leq j \leq q$. A node v of the full tree is in the known-hash tree if and only if for every node $v_i \neq v$ of type $i \geq 1$ on the path from v to the root the query (i, x_i, y_i) is in $\mathcal{Q}$ where x_i is the value of v_i and where y_i is the value of the outgoing edge of v_i on the path to v. It follows that if v is in the known-hash tree then so are all of its ancestors, so this is defines a valid (but possibly infinite) tree.

If a node v of type 0 is in the known-hash tree then the adversary knows the hash of the word $w_1 w_2 \cdots w_m$ where $w_1, \ldots, w_m$ are the values of the outgoing edges of the nodes of type 0 on the path from the root to v. This hash is in fact equal to the value of node v. One can also see that every node of type $i \geq 1$ has outdegree ≤ 1 in the known-hash tree, since for every value x_i there is only one y_i such that $\pi_i(x_i) = y_i$. However the outdegree of every node of type 0 is always N^s, since if a node of type 0 is in the known-hash tree then so, by definition, are all of its children. We will call the [illegible] of a node v of type 0 the number of outgoing edges from v that lie on a path to a node of type 0 further down the tree from v. The [illegible], or simply [illegible], is the restriction of the known-hash tree to nodes of type 0, where there is an edge from u to v in the reduced tree if and only if u is the first node of type 0 on the path from v to the root in the known-hash tree. Note that the outdegree of a node v in the reduced tree is equal to the reduced outdegree of v in the known-hash tree. One can define a natural bijection from the outgoing edges of v in the reduced tree to those outgoing edges of v in the known-hash tree that lie on a path to some node of type 0 further down. Using this bijection we can label in the natural way the edges of the reduced tree with values from $\{0,1\}^{sn}$. Then every path in the reduced tree corresponds to a word whose hash can be computed by the adversary, with the value of that hash being the value of the terminal node for that path. Thus the reduced tree gives a sort of digest of which hashes the adversary can compute[4] from the queries $\mathcal{Q}$.

For the attack, the adversary will make queries so as to grow the known-hash tree in a greedy fashion. It will make queries to $\pi_1, \ldots, \pi_k$ in cyclical order. When the adversary makes a query to π_i it will choose a value x_i that maximizes the number of terminal nodes of type i in the known-hash tree that have value x_i; that is, the adversary simply chooses the value such that there are a largest possible number of terminal nodes of type i with that value in the known-hash tree (here a [illegible] is a leaf of the known-hash tree). If there are no terminal nodes of type i, the adversary can make an arbitrary query to π_i. We assume the adversary makes kp queries in all, namely p queries to every permutation. Note that at any given query the known-hash tree could "blow up" and go to infinity; the number of added edges may be much larger than the number of terminal nodes.

This completes the description of the attack. We will now argue that, for q sufficiently large, the adversary has a good chance of obtaining a collision. First note that with kp greedy queries (not the ones we have described above), the pigeonhole argument shows that we can compute the value of the compression function on at least

$$N^{r+s}\left(\frac{p}{N}\right)^k \tag{1}$$

points in the domain $D = \{0,1\}^{r+s}$ of the compression function. This means that the average over the values $h \in \{0,1\}^{rn}$ of the number of points $w \in \{0,1\}^{sn}$ for which we can compute the value of the compression function on input $h \,\|\, w$ is

$$N^{r+s}\left(\frac{p}{N}\right)^k / N^r = N^s \left(\frac{p}{N}\right)^k \ . \tag{2}$$

On the other hand, the same average is approximated by the average outdegree of a node in the reduced tree after the adversary has carried out the above tree attack: every node corresponds to a value of h, and every outgoing edge corresponds to a value of w for which the output of the compression function on input $h \,\|\, w$ is known. The (heuristic) assumption underlying the tree attack is that for moderately large values of p, this outdegree average should approximate the average (2); after all, both the pigeonhole attack and the tree attack choose queries greedily. Then if (2) is moderately large, say equal to 2, we expect the reduced tree to have average outdegree close to 2. But [illegible] tree with average outdegree exceeding 1 must be infinite, and must also have unbounded width;

[4] The adversary may even know how to compute more hashes than those given from the reduced tree, for example if the function $g(h, w, y_1, \ldots, y_k)$ ignores some of the y_i's, making it not necessary to know their values. However since we are describing an attack and not a proof of security, this is irrelevant.

thus the reduced tree has blown up to infinity and we can find a collision by the pigeonhole principle (and even find a collision at the same level of the tree—meaning a collision of equal-length strings—because the width is unbounded).

To be more concrete, say that we choose $p = q/k$ large enough that

$$N^s \left(\frac{p}{N}\right)^k \geq 2 \tag{3}$$

Then one would expect that with some constant probability close to 1, but say with at least probability 1/2, the tree attack yields a reduced tree of average outdegree exceeding 1. Then the reduced tree has blown up to infinity and we hold a collision. This would give us an attack with probability of success 1/2. The cost of the attack would be $q = kp$ where

$$p = \left\lceil 2^{1/k} N^{1-s/k} \right\rceil \approx 2^{1/k} N^{1-s/k} , \tag{4}$$

which is to say $q \approx k\, 2^{1/k}\, N^{1-\alpha} \leq 2k\, N^{1-\alpha}$, because $\alpha = s/k$. This is an improvement on the bound for the pigeonhole-birthday attack since we expect k to be significantly smaller than n.

Theorem 7. $H\colon \{0,1\}^* \to \{0,1\}^{rn}$ α k $N = 2^n$

$q \approx 2k\, N^{1-\alpha}$

$\geq 1/2$ □

Most iterated hash functions have $s = 1$, in which case $k = k/s = 1/\alpha = \beta$ and the bound of Theorem 7 can be rewritten as $2\beta\, N^{1-\alpha}$; this is the version of the bound used for the numerical examples of Fig. 3. Note that for $\alpha = k = 1$, the case considered by Black [1], the tree attack gives a bound of $q = 2$ queries. This may seem seem small, but as Black note, any construction in which for any $h, x_1 \in \{0,1\}^n$ there is some $w \in \{0,1\}^n$ such that $x_1 = f_1(h, w)$ can indeed be broken in two queries, using the same argument as for the tree attack (in such a construction, the tree trivially blows up to infinity after just two queries, with uniform reduced outdegree of 2). Moreover, natural constructions will have this feature since it seems undesirable for the function $f_1(h, \cdot)$ to contain collisions (as a function from $\{0,1\}^n$ to $\{0,1\}^n$). However, for constructions that are artificially designed to hold off the attack, the bound $2kN^{1-\alpha}$ may be overly optimistic when it is very small (but in this case one does not much mind being off).

New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5

Lei Wang, Kazuo Ohta, and Noboru Kunihiro

The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585, Japan
{wanglei,ota,kunihiro}@ice.uec.ac.jp

Abstract. At Crypto '07, Fouque, Leurent and Nguyen presented full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5, by extending the partial key-recovery attacks of Contini and Yin from Asiacrypt '06. Such attacks are based on collision attacks on the underlying hash function, and the most expensive stage is the recovery of the so-called outer key. In this paper, we show that the outer key can be recovered with near-collisions instead of collisions: near-collisions can be easier to find and can disclose more information. This improves the complexity of the FLN attack on HMAC/NMAC-MD4: the number of MAC queries decreases from 2^{88} to 2^{72}, and the number of MD4 computations decreases from 2^{95} to 2^{77}. We also improved the total complexity of the related-key attack on NMAC-MD5. Moreover, our attack on NMAC-MD5 can partially recover the outer key without the knowledge of the inner key, which might be of independent interest.

Keywords: HMAC, NMAC, key-recovery, MD4, MD5, differential attack, near-collision.

1 Introduction

Many cryptographic schemes and protocols use hash functions. Their actual security might need to be reassessed, in light of the seminal work by Wang [12,13,14,15] on finding collisions on hash functions from the MD4 family. This paper deals with key-recovery attacks on HMAC and NMAC using differential techniques. HMAC and NMAC are hash-based message authentication codes proposed by Bellare, Canetti and Krawczyk [1]. HMAC has been implemented in widely used protocols including SSL, TLS, SSH, and IPsec. The construction of HMAC/NMAC is based on a keyed hash function. Let H be an iterated Merkle-Damgård hash function, which defines a keyed hash function H_k by replacing the IV with the key k. Then HMAC and NMAC are defined as:

$$\mathrm{HMAC}_k(M) = H(\bar{k} \oplus opad || H(\bar{k} \oplus ipad || M));$$
$$\mathrm{NMAC}_{k_1,k_2}(M) = H_{k_1}(H_{k_2}(M)),$$

where M is the input message, k and (k_1, k_2) are the secret keys of HMAC and NMAC respectively, $\bar{k}$ means k padded to a single block, $||$ means concatenation, and

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 237–253, 2008.

opad and *ipad* are two one-block length constants. NMAC is the theoretical foundation of HMAC: HMAC_k is essentially the same as $\text{NMAC}_{H(\bar{k}\oplus opad),H(\bar{k}\oplus ipad)}$, except with a change in the length value included in the padding. In [1,2], the security proof was first given for NMAC, and then extended to HMAC. Attacks on NMAC can usually be adapted to HMAC, except in the related-key setting. Hereafter, k_1 and k_2 (for HMAC: $H(\bar{k} \oplus opad)$ and $H(\bar{k} \oplus ipad)$ with the appropriate changes in the padding) are referred to as the outer key and the inner key, respectively. The corresponding hash functions of k_1 and k_2 are referred to as the outer hash function and the inner hash function, respectively.

The security of HMAC and NMAC

The security of HMAC /NMAC has been carefully analyzed by its designers [1,2]. It has been proved that NMAC is a pseudo-random function family (PRF) under a single assumption: (1) compression function of the keyed hash function is a PRF. The proof for NMAC has been extended to HMAC by an additional assumption: (2) the key derivation function in HMAC is a PRF. However, if the underlying hash function is weak (such as MD4 and MD5), the above proofs may not apply.

There are three types of attacks [4,5,6,8,9] on HMAC/NMAC:

- [illegible] distinguish HMAC/NMAC from a random function.
- [illegible] compute a valid MAC for a random message.
- [illegible] compute a valid MAC for any given message.

We focus on universal forgery attacks, by trying to recover the secret keys k_1 and k_2, like in previous work [4,5,9]. Contini and Yin [4] proposed partial key-recovery attacks on HMAC/NMAC instantiated with MD4, MD5[1], SHA-0 and step-reduced SHA-1. Their attacks can only recover the inner key k_2, which is insufficient for a universal forgery attack. Fouque, Leurent and Nguyen [5] presented the first full-key attack on HMAC/NMAC-MD4, by proposing an outer-key recovery attack. They also extended the attack of [4] into a full key-recovery attack on NMAC-MD5 in the related-key setting: this attack was independently found by Rechberger and Rijmen [9], who also proposed a full key-recovery attack in the related-key setting on NMAC with SHA-1 reduced to 34 steps. These full key-recovery attacks first apply the attack of [4] to recover the inner key k_2, then use additional MAC queries to derive several bits of the outer key k_1, and finally the rest of the outer key is obtained by the exhaustive search using offline hash computations. Recovering the outer key is so far the most expensive stage.

Our contributions

We propose new outer-key recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5[2], which leads to full key-recovery attacks by using the inner-key attacks of [4]. Compared to previous work by Fouque [illegible] [5], the main novelty is the use of [illegible] instead of collisions. Recall that a near-collision is a pair

[1] The attack on NMAC-MD5 is a related-key attack, and therefore does not apply to HMAC-MD5.

[2] Our attack on NMAC-MD5 is in the related-key setting, like [5,9].

of distinct messages whose hash values are almost the same, differing only by a few bits (see [7]): our near-collisions are based on a local collision at some intermediate step of the compression function, which significantly simplifies the difference propagation in the last few steps. Our attacks can be sketched as follows. We call the MAC oracle on exponentially many messages chosen in such a way that we can expect to find near-collisions in the outer hash function. By observing the shape of the near-collisions obtained, we are able to derive certain bits of the final values of the four 32-bit intermediate values a, b, c, d of the outer hash function. This discloses a few bits of the outer key k_1, since each 128-bit MAC value is exactly $(k_a+a, k_b+b, k_c+c, k_d+d)$ because MD4 and MD5 use the Davies-Meyer mode, where k_1 is decomposed as four 32-bit variables k_a, k_b, k_c and k_d.

The cost of our attacks is summarized in Table 1. In the case of HMAC/NMAC-MD4, near-collisions are easier to find and disclose more information, which allows to considerably improve the FLN attack [5] in both the number of MAC queries and the number of offline MD4 computations. In the case of NMAC-MD5, compared to the FLN-RR attack [5,9], total complexity is decreased. Moreover, we note that our attack can partially recover the outer key without the knowledge of the inner key k_2, which might be of independent interest.

Table 1. Comparison with previous work

Universal forgery attack		previous result	our new result
HMAC-MD4 NMAC-MD4	Online queries	2^{88} [5]	2^{72}
	Offline MD4 computations	2^{95} [5]	2^{77}
	Total complexity	2^{95}	2^{77}
NMAC-MD5 related-key setting	Online queries	2^{51} [5,9]	2^{75}
	Offline MD5 computations	2^{100} [5,9]	2^{75}
	Total complexity	2^{100}	2^{76}

Organization of the paper
Section 2 reviews background and related work. In Section 3, we explain the advantages of our attacks compared to previous work. In Sections 4 and 5, we present in details our attacks on HMAC/NMAC-MD4 and NMAC-MD5. Finally, we conclude and give open problems in Section 6.

2 Background and Notation

2.1 Description of MD5 and MD4

There is no standard notation for the description of MD5 and MD4. In this paper, we adopt a notation similar to that of [4].

MD5 and MD4 have the Merkle-Damgård structure and output a 128-bit hash value. First, the input message is padded to be the multiple of 512 bits: add '1' in the tail of the input message; add '0's until the bit length becomes

448 modulo 512; add the length of input message (before padding) to the last 64 bits. Then the padded message M is divided into 512-bit messages $M = (M_0, M_1, \ldots, M_{n-1})$. The 128-bit IV is represented as H_0 (which is the secret key in the keyed hash function). The compression function is first applied on M_0 and H_0 as input, which outputs a 128-bit value H_1. By iterating over all the message blocks M_i, we obtain a final 128-bit value H_n, which is defined to be the hash value of M.

Compression function of MD5

The compression function takes a 512-bit message block m and a 128-bit value H as input. First, m is divided into sixteen 32-bit values $(m_0, \ldots, m_{15})$, and H is divided into four 32-bit variables (a_0, b_0, c_0, d_0). The compression function consists of 64 steps, regrouped into four 16-step rounds. Each step is defined as follows:

$$\begin{aligned} &a_i = d_{i-1}, c_i = b_{i-1}, d_i = c_{i-1}, \\ &b_i = b_{i-1} + (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \qquad s_i, \end{aligned}$$

where m_k is one of $(m_0, \ldots, m_{15})$, the index k being given by a permutation of $\{0, \ldots, 15\}$ depending on the round, t is a constant defined in each round, $\quad s_i$ means a left-rotation by s_i bits, and f is a Boolean function depending on the round.

1R: $f(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
2R: $f(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$
3R: $f(X, Y, Z) = X \oplus Y \oplus Z$
4R: $f(X, Y, Z) = (X \vee \neg Z) \oplus Y$

The final output is $(a_0 + a_{64}, b_0 + b_{64}, c_0 + c_{64}, d_0 + d_{64})$, which means that MD5 uses the Davies-Meyer mode.

Compression function of MD4

The differences between MD5 and MD4 are the following:

- MD4 consists of 48 steps regrouped into three 16-step rounds.
- Each step is defined as: $b_i = (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \qquad s_i$, where m_k is given by different round permutations.
- In the 2nd round: $f(X, Y, Z) = (X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z)$.

2.2 Pseudo-collision of MD5

In [3], den Boer and Bosselaers found a pseudo-collision on the compression function of MD5 of the following form:

$$\text{MD5}(IV,\ M) = \text{MD5}(IV',\ M)$$

Here, the one-block message M is the same, and only the IVs are different. The total probability of their pseudo-collision is 2^{-46}, provided that IV and IV' satisfy the following relations:

- $\Delta IV = (IV \oplus IV') = (0x80000000, 0x80000000, 0x80000000, 0x80000000)$;
- If we decompose the IV as four 32-bit variables (a_0, b_0, c_0, d_0), then the MSBs of b_0, c_0 and d_0 must be the same.

In the rest of this paper, the difference ΔIV of their pseudo-collision will be denoted by Δ^{MSB}, and this pseudo-collision will be referred to as the dBB pseudo-collision.

2.3 Recovering the Inner Key of HMAC/NMAC-MD4

We recall the differential attack of Contini and Yin [4] to recover the inner key:

1. Determine a message difference ΔM and a differential path DP for a collision attack on MD4. Let n be the number of sufficient conditions.
2. Generate a random one-block message M, and send both M and $M + \Delta M$ to the HMAC/NMAC oracle until one pair of messages $(M_1, M_1 + \Delta M)$ collides. Since the number of sufficient conditions is n, such a pair $(M_1, M_1 + \Delta M)$ will be obtained after roughly 2^n pairs of messages are queried.
3. Recover the intermediate chaining variables (ICV) in step t of 1R of $H(k_2, M_1)$. This technique is one main contribution of the inner-key recovery attack of Contini and Yin [4]. For details, please refer to [4].
4. Derive the inner key k_2 by inverse calculation from the obtained ICV. This is easy since each step of MD4 is invertible. For instance, with MD4, if m_{t-1} and ICV in step t are known, ICV in step $t-1$ can be calculated as follows.

$$b_{t-1} = c_t, c_{t-1} = d_t, d_{t-1} = a_t,$$
$$a_{t-1} = (b_t \quad s_0) - m_{t-1} - f(c_t, d_t, a_t).$$

The related-key attack on NMAC-MD5 [4] is based on the same ideas. The attack exploits the freedom over the input messages, which explains why this attack is the most efficient attack known to recover the inner key k_2. However, for the outer hash function of HMAC/NMAC, the input message is the output of the inner hash function, for which there is much less freedom. This attack is therefore not well-suited to recover the outer key.

2.4 Recovering the Outer Key of HMAC/NMAC-MD4

We recall the differential attack of Fouque, Leurent and Nguyen [5] to recover the outer key:

1. Determine a message difference ΔM and a differential path DP for a collision attack on MD4 in such a way that the differential path has one sufficient condition depending on one bit of k_1. Let n be the number of sufficient conditions without counting the one on k_1.
2. Generate pairs of messages (M, M') satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$. This technique is detailed in Appendix A, which will be utilized in our own attack.

3. Send M and M' to the HMAC/NMAC oracle. Once roughly 2^n pairs of messages (M, M') are queried, if a collision is obtained, the outer key k_1 satisfies the sufficient condition. Otherwise, k_1 is very unlikely to satisfy the sufficient condition. So with 2^{n+1} queries, we will recover one bit of k_1.
4. Change ΔM and DP, and recover other bits of k_1.

The first two steps are the most important steps of the attack [5]. The main idea is to find a differential path with one sufficient condition on the outer key k_1. If k_1 satisfies the condition, a collision will be found with a suitable number of queries. Otherwise, no collision is likely to be found after the same number of queries. This will disclose bits of k_1.

However, if we divide the outer key k_1 as (k_a, k_b, k_c, k_d) for the computation of the outer MD4, then it turns out that such conditions can only be set on k_b and k_c, so the attack can not recover any of the bits of k_a and k_d.

3 Attacks on HMAC/NMAC with Near-Collisions

In this section, we give an overview of our new attacks on HMAC/NMAC based on near-collisions. A detailed description of the attacks will be given in respectively Section 4 for the MD4 case, and Section 5 for the MD5 case.

3.1 Overview

We first give an overview in the case of MD4. Thanks to [4], we can already assume that we know the inner key k_2 of HMAC/NMAC-MD4, and we want to recover the outer key k_1, which will be decomposed as four 32-bit variables k_a, k_b, k_c and k_d. Because MD4 uses the Davies-Meyer mode, we know that the 128-bit value of HMAC/NMAC-MD4 is exactly $(k_a + a, k_b + b, k_c + c, k_d + d)$, where a, b, c, d denote the final values of the four 32-bit intermediate values of the outer MD4.

The FLN attack [5] used an IV-dependent differential path for MD4 collisions, and derived bits of k_1 by observing whether or not collisions for the outer MD4 occurred. We will use a differential path for MD4 near-collisions which is independent of the IV, and we will collect near-collisions. These near-collisions are based on a local collision at some intermediate step of the MD4 compression function. Thanks to special properties of our differential path, we will be able to extract certain bits of (a, b, c, d), depending on the shape of the near-collision. Because of the Davies-Meyer mode, this will disclose certain bits of k_1.

Thus, the structure of our attack on HMAC/NMAC-MD4 is the following:

1. Determine a message difference ΔM and a differential path DP for a near-collision attack on MD4. Let n be the number of sufficient conditions.
2. Generate pairs of messages (M, M') satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$. We can use the FLN technique [5], described in Appendix A.
3. Send M and M' to the HMAC/NMAC-MD4 oracle. Once roughly 2^n pairs of messages (M, M') are queried, we obtain a near-collision.

4. Once a near-collision with (M, M') is obtained, we look at the shape of the near-collision: due to choice of our differential path, we know that certain shapes of near-collisions can only arise if certain bits of (a, b, c, d) are equal to 1 at the end of the computation of NMAC-MD4(M). This discloses bits of k_1 thanks to the Davies-Meyer mode.
5. Change ΔM and DP, and recover other bits of k_1.

Our related-key attack on NMAC-MD5 is based on similar ideas. We use the differential path of [3] associated to the dBB pseudo-collision. This differential path also gives rise to near-pseudo-collisions, that is, MD5(IV, M) and MD5(IV', M) only differ by a few bits. Of course, instead of calling the NMAC-MD5 oracle on random messages M and M' such that $H_{k_2}(M') = H_{k_2}(M) + \Delta M$, we will call the NMAC-MD5 oracle on a randomly chosen M with two related keys corresponding to Δ^{MSB}. Because this does not use the inner key k_2, we will thus be able to recover bits of k_1 without knowing k_2.

3.2 Features

We summarize the main features of our attacks, compared to [5,9]:

The HMAC/NMAC-MD4 case:

- Generating a near-collision requires much less queries than a collision. Compared to the FLN attack [5], the number of MAC queries is reduced to 2^{72} from 2^{88},
- Our MD4 near-collisions disclose more information than collisions. Indeed, we can recover bits of k_b, k_c and k_d, rather than just bits of k_b and k_c. Compared to the FLN attack [5], this discloses 51 bits of the outer key k_1, instead of only 22 bits. Hence, the number of offline MD4 computations is reduced to 2^{77} from 2^{95} (FLN attack decreased their offline complexity to 2^{95} from 2^{106} using some speeding up technique. Please refer to [5] for details.).

The NMAC-MD5 case:

- our attack does not require any control over the input messages, so our attack can partially recover the outer key k_1 without knowing the inner key k_2, unlike previous work. This might be of independent interest.
  We increase the number of online queries, but we can derive more information on the outer key: 63 bits of k_1 can be recovered, instead of only 28 bits [5,9]. There is no standard calculation method of the total complexity. We will follow that of [9]: the sum of the online complexity and the offline complexity. Finally we recovered 53 bits of k_1 in order to make the online and the offline complexity be equal: 2^{75}. The total complexity of MD5 computations is reduced to 2^{76} from 2^{100}.

4 New Key Recovery Attack on HMAC/NMAC-MD4

We now precisely describe our new outer-key recovery attack on HMAC/NMAC-MD4. Recall that the outer key k_1 is decomposed as (k_a, k_b, k_c, k_d).

Denote the final values (after 48 steps) of the 32-bit intermediate values of the outer MD4 as (a_{48}, b_{48}, c_{48}, d_{48}). Then the output of HMAC/NMAC-MD4 is: (h_a, h_b, h_c, h_d)=($k_a + a_{48}$, $k_b + b_{48}$, $k_c + c_{48}$, $k_d + d_{48}$). So we have the following relations when comparing two outputs of HMAC/NMAC-MD4:

$$\Delta h_a{=}\Delta a_{48},\ \Delta h_b{=}\Delta b_{48},\ \Delta h_c{=}\Delta c_{48} \text{ and } \Delta h_d{=}\Delta d_{48}.$$ [3]

As a result, we can detect the difference propagation in the last four steps of the outer MD4 from the final output values of HMAC/NMAC. Based on this weakness of HMAC/NMAC-MD4 due to the Davies-Meyer mode, we will obtain bit-values of a_{48}, c_{48} and d_{48}. This, in turn, will disclose bits of k_a, k_c and k_d.

Our attack has both online work and offline work. We will first describe our near-collision on MD4. Then, we will explain details of online work and offline work.

4.1 Near-Collisions on MD4

The main contribution of this paper is the use of near-collisions. Our near-collisions on MD4 are based on a local collision at step 29. We determine the message differences ΔM as $\Delta m_3{=}2^i$, that is, the messages only differ in m_3. The corresponding differential path is given in Appendix D. This differential path works for the cases $i = 3 \sim 5, 7 \sim 17, 20 \sim 25$: other values of i fail because of carry expansion.

The above near-collisions have the following properties:

- m_3 is used in step 45 of 3R. If the local collision in step 29 happens, the differences propagation in the last four steps will be significantly simplified.
- Because we use a local collision in step 29, we only need to consider the differential path until step 29. This reduces the number of sufficient conditions, and therefore the number of queries to obtain a near-collision.

4.2 Online Work: Obtaining Bit-Values of a_{48}, c_{48} and d_{48}

The procedure is as follows, where the message difference ΔM is $\Delta m_3 = 2^i$:

1. Generate pairs of messages (M, M') such that MD4(k_2, M')=MD4(k_2, M)+ ΔM. We adapt the technique proposed in [5], which is given in Appendix A.
2. Send such messages M and M' to the HMAC/NMAC-MD4 oracle to obtain any of the following three kinds of near-collisions:
   - Pairs (M_a^i, M_a^{i}') such that $\Delta h_a{=}2^{i+3}$, $\Delta h_d{=}{*}2^{i+12}$ and $\Delta h_c{=}{*}2^{i+23} \pm 2^{i+14} \pm 2^{i+15}$; [4]
   - Pairs (M_c^i, M_c^{i}') such that $\Delta h_a{=}2^{i+3}$, $\Delta h_d{=}{*}2^{i+12}$, $\Delta h_c{=}{*}2^{i+23} * 2^{i+14}$, and expected Δh_b; [5]

[3] If two values differ at the MSBs, there will exist error probability. We will ignore such situations because they do not happen in our attack.

[4] $*$ means that the sign does not matter, and $\pm 2^{i+14} \pm 2^{i+15}$ means that the signs of these two differences are the same.

[5] Δh_b consists of $\pm 2^{i+6} \pm 2^{i+7}$.

  - Pairs (M_d^i, M_d^{i}') such that: $\Delta h_a = 2^{i+3}$, $\Delta h_d = *2^{i+12}$ and $\Delta h_c = *2^{i+14} \pm 2^{i+23} \pm 2^{i+23}$.

3. Change the index i, and repeat steps 1 and 2 until all values of i are used.

First, let us observe that the above near-collisions are very likely to come from our differential path. Indeed, the shape of our near-collisions impose fixed differences on three 32-bit words, so a pair (M, M') chosen uniformly at random would give such a near-collision with probability 2^{-96}. However, our pairs (M, M') chosen in step 1 have a much higher probability 2^{-64} to near-collide.[6]

We now claim that the messages obtained above with near-collisions satisfy the following conditions on the final values of the intermediate values of the outer MD4:

$$M_a^i\text{: } a_{48,i+3} = 1;\ M_c^i\text{: } c_{48,i+3} = 1;\ M_d^i\text{: } d_{48,i+3} = 1.$$

For instance, consider the case of M_a^i. Because of the near-collision, the difference propagation in 3R only exists in the last four steps. At step 47, the variable generated is c_{48}. And input differences only exist in a_{48} and d_{48}: $\Delta a_{48} = 2^{i+3}$ and $\Delta d_{48} = *2^{i+12}$. Since the number of the bits of the left rotation is 11, both $\pm 2^{i+14}$ and $\pm 2^{i+15}$ of Δc_{48} must be caused by 2^{i+3} of Δa_{48}. Such a difference propagation can not happen if there does not exist a carry during the calculation $a_{48}+2^{i+3}$, so the probability of $a_{48,i+3}=1$ is 1. With a similar reasoning, the messages M_c^i and M_d^i satisfy $c_{48,i+3} = 1$ and $d_{48,i+3} = 1$, respectively.

Finally, we can obtained near-colliding messages M_a^i such that $a_{48,i+3}=1$ for $i=3 \sim 5$, $7 \sim 15$, $20 \sim 25$: other values of i fail because of carry expansion. In total, there are 18 near-colliding messages M_a^i, which can disclose values of $k_{a,i+3}$. Details are shown in section 4.3. So we can recover 18 bit-values of k_a by online work. Similarly, k_c and k_d are also partially recovered by online work. Near-colliding messages M_c^i and M_d^i are obtained for $i = 3 \sim 5$, $9 \sim 17$, $20 \sim 23$ and $i = 3 \sim 5$, $9 \sim 17$, $21 \sim 25$ respectively. So 16 bit-values of k_c and 17 bit-values of k_d, corresponding $k_{c,i+3}$ and $k_{d,i+3}$ of M_c^i and M_d^i respectively, can be recovered. In total, 51 bits of the outer key k_1 are recovered by the online work.

4.3 Offline Work: Recovering k_a, k_c and k_d

The way to recover k_a, k_c and k_d is the same. We will pick k_a as an example to explain the details:

1. Guess the values of $k_{a,i}$ for $i = 0 \sim 5$, 9, $19 \sim 22$, $29 \sim 31$: the index i that we fail obtaining M_a^{i-3}. These bit-values of k_a will be recovered by the offline exhaustive search. The total number of possibilities is 2^{14}.
2. Calculate other bits of k_a from the least significant to the most significant bits using M_a^i. First, the 6-th bit of k_a will be calculated using M_a^3.
   - [illegible] $k_{a,6}$ [illegible] $k_{a,5\sim0}$ [illegible] $h_{a,5\sim0}$ [illegible] $k_{a,5\sim0} > h_{a,5\sim0}$ [illegible] k_a a_{48} [illegible]

[6] Details are shown in section 4.4.

k_a a_{48} … $a_{48,6}$ … $h_{a,6}$
… $k_{a,6}$ …

Then, the 7-th bit will be derived from M_a^4. Then the 8-th bit, and so on. Finally, all other bits of k_a will be recovered.

By a similar process, all the bits of k_c and k_d will be recovered.

4.4 Complexity Analysis

As explained in section 4.2, we can obtain 18 bits, 17 bits and 16 bits of k_a, k_d and k_c using M_a^i, M_d^i and M_c^i, respectively. Totally 51 bits of k_1 are recovered by the online work, so the complexity of the offline exhaustive search is 2^{77} (2^{128-51}) MD4 computations.

Now we analyze the complexity of online work. This depends on the probability of the specified shape of near-collision, which can be regarded as two parts: probability of near-collision and that of specified difference propagation in the last four steps. The probability of our near-collisions is 2^{-60} since there are in total 60 conditions of differential path. The probabilities of difference propagation in the last four steps of outer MD4 are shown in Appendix B. One pair (M_a^i, M_a^{i}'), (M_c^i, M_c^{i}'), and (M_d^i, M_d^{i}') can be obtained with a probability $2^{-60} \times 1 \times \frac{2}{3} \times \frac{1}{9}$ (greater than 2^{-64}), $2^{-60} \times \frac{2}{3} \times \frac{4}{9} \times \frac{1}{4}$ (greater than 2^{-64}), and $2^{-60} \times \frac{2}{3} \times \frac{1}{9}$ (greater than 2^{-64}) respectively: one above pair can be obtained with roughly 2^{66} queries. As a result, the total online complexity is 51×2^{66} (less than 2^{72}) queries.

Experiment
It is impossible to carry out the real experiment. Instead, we separate the experiment to two parts:

- Confirm the correctness of DP: an example is shown in Appendix C.
- Confirm the correctness of key recovery technique by only focusing on the last four steps of outer MD4: the intermediate variables at step 44 and the message m_3 are randomly generated.

5 New Key Recovery Attack on NMAC-MD5

Similarly with MD4 case, we can detect the difference propagation in the last four steps of the outer MD5 from the final output values of HMAC/NMAC-MD5. It seems that our near-collision attack can be extended to HMAC/NMAC-MD5. However, we have not found suitable message difference and differential path for near-collision on MD5. Thanks to dBB pseudo-collision, where the difference propagation in the last four steps of the outer MD5 is very simple, we will be able to obtain bit-values of the intermediate values (after 64 steps) in the outer MD5 by detecting the shape of near-pseudo-collision or pseudo-collision. This, in turn, will disclose the outer key k_1.

In this section, we will explain the details of our outer-key recovery attack on NMAC-MD5 in the related-key setting: the attacker obtains $\mathrm{MD5}_{k_1}(\mathrm{MD5}_{k_2}(M))$

and $\mathrm{MD5}_{k'_1}(\mathrm{MD5}_{k_2}(M))$ denoted as NMAC and NMAC' respectively hereafter; k_1 and k'_1 satisfy Δ^{MSB} defined in section 2.2. Recall that k_1 is decomposed as (k_a, k_b, k_c, k_d). Denote the intermediate variables (after 64 steps) in the outer MD5 as $(a_{64}, b_{64}, c_{64}, d_{64})$. Then the output of NMAC-MD5 is: $(h_a, h_b, h_c, h_d) = (k_a + a_{64}, k_b + b_{64}, k_c + c_{64}, k_d + d_{64})$.

Our new outer-key recovery attack consists of online work and offline work. The online work partially recovers k_a and k_c without knowledge of the inner key k_2, which might be of independent interest. The offline work is just the exhaustive search, where the inner key is necessary. We will first describe near-pseudo-collision on MD5. Then we will explain details of the online work. Since the offline work is just the exhaustive search, we will omit it.

5.1 Near-Pseudo-collision on MD5

According to dBB pseudo-collision, once a local collision happens at step 63, the shape of near-pseudo-collision will depend on $a_{64,31}$ and $c_{64,31}$:

- if $a_{64,31} = c_{64,31}$: collision happens;
- if $a_{64,31} \neq c_{64,31}$: the final output differences are $\Delta h_a = 0$, $\Delta h_b = \pm 2^{20}$, $\Delta h_c = 0$ and $\Delta h_d = 0$.

So we can obtain the relation between $a_{64,31}$ and $c_{64,31}$ by detecting the shape of near-pseudo-collision.[7]

5.2 Online Work: Recovering $k_{a,31\sim 30}$ and $k_{c,31\sim 30}$

The procedure is as follows:

1. Generate messages randomly and send them to NMAC and NMAC' to obtain near-pseudo-colliding messages $\{M\}$, regrouped depending on the values of $h_{a,30}$ and $h_{c,30}$:
   - $\{M_0\} : h_{a,30} = 0$ and $h_{c,30} = 0$;
   - $\{M_1\} : h_{a,30} = 0$ and $h_{c,30} = 1$;
   - $\{M_2\} : h_{a,30} = 1$ and $h_{c,30} = 0$;
   - $\{M_3\} : h_{a,30} = 1$ and $h_{c,30} = 1$.
2. Determine relation between $k_{a,31}$ and $k_{c,31}$ based on each element of each sub-group utilizing the following tool:
   [illegible] $k_a + a_{64} / k_c + c_{64}$ [illegible] $h_{a,30}/h_{c,30} = 0$ [illegible]
   [illegible]
3. Check the results of step 2 for each sub-group. There should be only one sub-group that all elements have the same result, which will disclose $k_{a,31\sim 30}$ and $k_{c,31\sim 30}$ as follows:
   the result of step 2 is the real relation between $k_{a,31}$ and $k_{c,31}$;
   $k_{a,30} = 1 - h_{a,30}$; $k_{c,30} = 1 - h_{c,30}$.

7 Hereafter, we regard pseudo-collision as a special kind of near-pseudo-collision just for simplicity.

First we will explain why the relation between $k_{a,31}$ and $k_{c,31}$ can be determined at step 2: the above tool determines the carry influence from bit 30 to 31 during $k_a + a_{64}/k_c + c_{64}$; the shapes of near-pseudo-collisions show the relation between $a_{64,31}$ and $c_{64,31}$; the relation between $h_{a,31}$ and $h_{c,31}$ is easy to check. Pick one pseudo-colliding element $m \in \{M_0\}$ as an example. We can obtain that $a_{64,31} = c_{64,31}$; there exists a carry from bit 30 to 31 during $k_a + a_{64}/k_c + c_{64}$. Consequently, the relation between $k_{a,31}$ and $k_{c,31}$ is determined as follows:

$$h_{a,31} = h_{c,31} \Rightarrow k_{a,31} = k_{c,31};$$
$$h_{a,31} \neq h_{c,31} \Rightarrow k_{a,31} \neq k_{c,31}.$$

Then we will explain why only one sub-group does not have different results at step 2. This is because of the utilized tool. The error probability of the tool depends on the relation between $k_{a/c,30}$ and $h_{a/c,30}$.

- $k_{a/c,30} = h_{a/c,30}$: error probability is $\frac{1}{2}$. For example, if both values are 0, according to the tool, we will assume that there is always a carry from bit 30 to 31. However, in fact the carry influence depends on the value $a/c_{64,31}$: carry exists if $a/c_{64,30} = 1$, and no carry if $a/c_{64,30} = 0$. Since the value of $a/c_{64,30}$ is random, the error probability is $\frac{1}{2}$.
- $k_{a/c,30} \neq h_{a/c,30}$: error probability is 0. For example, if $k_{a/c,30} = 0$ and $h_{a/c,30} = 1$, we can obtain that $k_{a/c,30\sim 0} < h_{a/c,30\sim 0}$, so there will be no carry with probability 1, which is the same with the tool.

So only the sub-group satisfying $k_{a/c,30} \neq h_{a/c,30}$ should be without error. In other words, all elements of this sub-group have the same result at step 2. This also explains the way we recover $k_{a/c,31\sim 30}$ at step 3.

5.3 Online Work: Recovering Other Bits of k_a and k_c

Since the way of recovering k_a is exactly the same with that of recovering k_c, we will pick k_a as an example in this section. The value of k_a is recovered from the most significant to the least significant bit. Suppose bits $k_{a,30\sim(i+1)}$ $(0 \leq i \leq 29)$ have been already obtained. The following procedure shows how to recover $k_{a,i}$.

1. Randomly generate messages and send them to the two NMACs until one message M_1 obtained satisfying the following three conditions:
   a) near-pseudo-collision happens;
   b) $h_{a,j}$=$k_{a,j}$ $(i+1 \leq j \leq 30)$;
   c) $h_{c,30} \neq k_{c,30}$.
2. Determine the carry influence from bit i to $i+1$ during $k_a + a_{64}$, where a_{64} is the intermediate value (after 64 steps) of the outer MD5 of MD5$_{k_1}$(MD5$_{k_2}(M_1)$).
3. Determine the value of $k_{a,i}$ by the result of step 2.
   -Carry: $h_{a,i}$=1 $\Rightarrow$ $k_{a,i}$=1;
   $h_{a,i}$=0 $\Rightarrow$ repeat steps 1 and 2.
   -No carry: $h_{a,i}$=0 $\Rightarrow$ $k_{a,i}$=0;
   $h_{a,i}$=1 $\Rightarrow$ repeat steps 1 and 2.

First, we can easily obtain the carry influence from bit 30 to 31 during $k_a + a_{64}$ based on conditions a) and c): condition a) guarantees that the relation between $a_{64,31}$ and $c_{64,31}$ can be determined; condition c) guarantees that the carry influence from bit 30 to 31 can be determined during $k_c + c_{64}$.

Then, we will obtain the carry influence from bit i to $i+1$ based on condition b): condition b) guarantees that the carry influence from bit i to $i+1$ and that from bit 30 to 31 are the same during $k_a + a_{64}$.

Finally, we will recover the value of $k_{a,i}$: if there exists a carry from bit i to $i+1$ and $h_{a,i} = 1$, then $k_{a,i} = 1$with probability 1; if there does not exist a carry from bit i to $i+1$ and $h_{a,i} = 0$, then $k_{a,i} = 0$with probability 1;

5.4 Complexity Analysis

Near-pseudo-collision is with a rough probability 2^{-45} since there are in total 45 conditions until step 63 according to dBB pseudo-collision on MD5.

Complexity of recovering $k_{a,31\sim30}$ **and** $k_{c,31\sim30}$
As explained in section 5.2, the error probability of other sub-groups is $\frac{1}{2}$. So we need to generate four elements for each sub-group. To guarantee the attack will succeed, we will totally generate 32 elements for $\{M\}$. The complexity will be $32 \times 2^{46} = 2^{51}$ queries.

Complexity of recovering $k_{a,i}$ **and** $k_{c,i}$ $(0 \le i \le 29)$
Considering the complexity of recovering $k_{a,i}$ is the same with that of recovering $k_{c,i}$, we will pick $k_{a,i}$ as an example.

In section 5.3, it needs $2^{46} \times 2^{30-(i+1)+1} \times 2 = 2^{77-i}$ queries to obtain one message satisfying conditions a), b) and c) in step 1. According to steps 2 and 3, we might repeat step 1 twice. So totally the complexity is $2 \times 2^{77-i} = 2^{78-i}$ queries.

There is no standard calculation method of the total complexity. We will follow that of [9], which is the sum of the online and the offline complexity. If we will recover bits of $k_{a,30\sim i}$ and $k_{c,30\sim i}$, with roughly 2^{80-i} queries, the value of i should make the online and the offline complexity be equal: $2^{80-i} = 2^{128-(31-i)\times 2-1} \Rightarrow i = 5$. As a result, we will recover $k_{a,30\sim5}$, $k_{c,30\sim5}$ and the relation between $k_{a,31}$ and $k_{c,31}$. The online complexity is less than 2^{75} queries, and the offline complexity is 2^{75} MD5 computations.[8]

Experiment
It is impossible to carry out the real experiment. Similarly with HMAC/NMAC-MD4 case, we only focus on the last 4 steps of outer MD5, so we will randomly generate the intermediate variables at step 60 and messages m_2 and m_4.

6 Conclusion

This paper proposed new outer-key recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5 (with related-key setting).

[8] For the offline MD5 computations, we will assume the inner key k_2 has been obtained by the inner-key recovery attack of Contini and Yin [4].

So far, no key-recovery attack has been published on HMAC/NMAC-MD5 without related-key setting. There are two reasons: (1) the inner-key recovery attack of Contini and Yin [4] can not succeed because all differential paths published so far have more than 128 sufficient conditions; (2) Wang et al.'s collision attack on MD5, multi-block collision, can no be used for the outer-key recovery attack, because the input message of the outer MD5 is the hash values of the inner MD5, just one-block length.

Our near-collisions may solve the second problem, since our near-collisions are only one-block length. Here we focus on the outer-key recovery attack, and assume that the inner key has been obtained. Moreover, our near-collisions are easier to be obtained than collisions, only counting sufficient conditions until some intermediate step where a local collision happens.

As explained above, once the number of sufficient conditions of near-collision is less than 128, outer-key recovery attack might be a real attack on HMAC/NMAC-MD5 without related-key setting.

Acknowledgements

We would like to thank Phong Q.Nguyen for improving our paper and anonymous reviewers for helpful comments.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
3. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
4. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
5. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
6. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0, and SHA-1. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
7. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
8. Rechberger, C., Rijmen, V.: Note on Distinguishing, Forgery and Second Preimage Attacks on HMAC-SHA-1 and a Method to Reduce the Key Entropy of NMAC. Cryptology ePrint Archive, Report, 2006/290 (2006)

9. Rechberger, C., Rijmen, V.: On Authentication with HMAC and Non-Random Properties. In: Dietrich, S., Dhamija, R. (eds.) Financial Cryptography 2007. LNCS, vol. 4886, pp. 39–57. Springer, Heidelberg (2007)
10. Rivest, R.L.: The MD4 Message-Digest Algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
11. Rivest, R.L.: The MD5 Message Digest Algorithm. Request for Comments (RFC 1321), Network Working Group (1992)
12. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
13. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
14. Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
15. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

A FLN Attack: Generating Pairs of Messages (M, M') That $H_{k_2}(M')=H_{k_2}(M)+\blacktriangle M$ Efficiently

In [5], Fouque . . proposed an efficient way to generate pairs of messages (M, M') satisfying $H_{k_2}(M')=H_{k_2}(M)+\Delta M$.[9] This technique works on hash functions that have the Merkle-Damgård structure. The procedure is as follows:

1. Generate one pair of one-block length messages (M_1, M_1') satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$ by birthday attack, where padding is not considered. Since the output of MD4 is 128-bit length, (M_1, M_1') will be obtained after roughly 2^{64} MD4 computation.
2. (M_1, M_1') will be extended to a family of two-block pair messages such that $H_{k_2}(M_1||M_2) = H_{k_2}(M_1'||M_2') + \Delta M$. The length of M_2 and M_2' must be no longer than 447 bits because of the padding rule.

   Selecting M_2 and M_2'
   Denote $H_{k_2}(M)$ and $H_{k_2}(M')$ as h_1 and h_1', respectively. we will obtain that $H_{k_2}(M_1||M_2) = H_{h_1}(M_2)$ and $H_{k_2}(M_1'||M_2') = H_{h_1'}(M_2')$. Denote intermediate chaining variables after 48 steps as ICV_{48}. $\text{MD4}_{h_1}(M_2) = h_1 + \text{ICV}_{48}$. Similarly, $\text{MD4}_{h_1'}(M_2') = h_1' + \text{ICV}_{48}'$. Since $h_1' = h_1 + \Delta M$, if $\text{ICV}_{48}' = \text{ICV}_{48}$, $\text{MD4}_{h_1'}(M_2') = \text{MD4}_{h_1}(M_2) + \Delta M$, so $\text{MD4}_{k_2}(M_1||M_2) = \text{MD4}_{k_2}(M_1'||M_2') + \Delta M$. As explained above, M_2 and M_2' should satisfy that $\text{ICV}_{48} = \text{ICV}_{48}'$. Such pair M_2 and M_2' can be obtained utilizing Wang . . ' collision attack on MD4. Please refer to [5] for more details.

[9] As shown in section 2.4, ΔM is determined differences of inner hash values instead of $M' - M$.

B Probabilities of Difference Propagation in 3R

If near-collision happens, and the message difference ΔM is $\Delta m_3 = 2^i$.

- Δa_{48}=2^{i+3}: the probability is 1 except that bit i or $i+3$is MSB. During our attack, $i \leq 25$, so $i+3 \leq 28$.
- Δd_{48}=$*2^{i+12}$: the probability can be regarded as $\frac{2}{3}$. Δd_{48} depends on the bit carry expansion of Δa_{48} because f works bit-independently. f is XOR.
  No carry with probability $\frac{1}{2}$: Δd_{48}=$*2^{i+12}$ with probability 1.
  1-bit carry with probability $\frac{1}{4}$: Δd_{48}=$*2^{i+12}$ with probability $\frac{1}{2}$.
  2-bit carries with probability $\frac{1}{8}$: Δd_{48}=$*2^{i+12}$ with probability $\frac{1}{4}$.
  $$\vdots$$
  So the probability is almost $\frac{1/2}{1-1/4}$=$\frac{2}{3}$.

Δc_{48}=$*2^{i+23} \pm 2^{i+14} \pm 2^{i+15}$: Similarly with analysis above, $*2^{23}$ of Δc_{48} is with probability $\frac{2}{3}$. $\pm 2^{i+14} \pm 2^{i+15}$ of Δc_{48} is with probability $\frac{1}{6}$. Totally, the probability is $\frac{2}{3} \times \frac{1}{6} = \frac{1}{9}$.

Δc_{48}=$*2^{i+23} * 2^{i+24}$: similarly with analysis above, the probability is $\frac{2}{3} \times \frac{2}{3}$=$\frac{4}{9}$.

Δc_{48}=$*2^{i+14} \pm 2^{i+23} \pm 2^{i+24}$: similarly with analysis above, the probability is $\frac{2}{3} \times \frac{1}{6}$ =$\frac{1}{9}$.

$\pm 2^{i+6} \pm 2^{i+7}$ of Δb_{48}: the probability of Δc_{48} with a carry is $\frac{1}{2}$, and the probability that Δf consists of $\pm 2^{i+23} \pm 2^{i+24}$ is $\frac{1}{2}$. Totally, the probability is $\frac{1}{4}$.

C An Example of Near-Collision on HMAC/NMAC-MD4

In order to confirm the correctness of our differential path of near-collision on MD4, we will provide an example in Table 2. The messge difference is $\Delta m_3 = 2^3$.

Table 2. An example of near-collision

Outer key k_2	$k_a = 0xae23667d$; $k_b = 0x9ae8ba3c$; $k_c = 0x3775447e$; $k_d = 0x9614f6dc$
Near-colliding messages	$m_0 = 0x4bb5f397$; $m_1 = 0x9a645f8a$; $m_2 = 0x7f3529c4$; $m_3 = 0x1e7b831\bar{7}$
(output of the inner MD4)	$m'_0 = 0x4bb5f397$; $m'_1 = 0x9a645f8a$; $m'_2 = 0x7f3529c4$; $m'_3 = 0x1e7b831f$
Step 29 of the outer MD4	$a_{29} = 0x84f021a1$; $b_{29} = 0x89f4c2d8$; $c_{29} = 0x62dbbc57$; $d_{29} = 0x76bdb3a6$

D DP and SCs of Near-Collision on MD4

The shown DP and SC is for Δm_3=2^3. DP and SC for other cases can be derived from this one by rotating all the bit differences and bit conditions. Cases $i = 3 \sim 5$, $7 \sim 17$, $20 \sim 25$ succeeds.

Table 3. DP and SCs

Step i	Shift s_i	Δm_{i-1}	Δb_i Numerical difference	Δb_i Sufficient conditions
1	3			
2	7			
3	11			$b_{3,22} = b_{2,22}$
4	19	2^3	2^{22}	$b_{4,22} = 0$
5	3			$b_{5,22} = 0$
6	7			$b_{6,22} = 1$
7	11			$b_{7,9} = b_{6,9}$
8	19		2^9	$b_{8,9} = 0$
9	3			$b_{9,9} = 0$
10	7			$b_{10,9} = 1$
11	11			$b_{11,28} = b_{10,28}$
12	19		2^{28}	$b_{12,28} = 0$
13	3			$b_{13,28} = 0$
14	7			$b_{14,28} = 1$
15	11			$b_{15,15} = b_{14,15}$
16	19		2^{15}	$b_{16,15} = 0$
17	3			$b_{17,15} = b_{15,15}$
18	5			$b_{18,15} = b_{17,15}$
19	9			$b_{19,28} = b_{18,28}$, $b_{19,29} \neq b_{18,29}$, $b_{19,30} = b_{18,30}$
20	13		2^{28} $(28 \sim 30)$	$b_{20,0} = b_{19,0}$, $b_{20,28\sim29} = 1$, $b_{20,30} = 0$
21	3		-2^0	$b_{21,0} = 1$, $b_{21,28\sim30} = b_{19,28\sim30}$
22	5			$b_{22,0} = b_{20,0}$, $b_{22,28\sim30} = b_{21,28\sim30}$
23	9			$b_{23,0} = b_{22,0}$, $b_{23,9} = b_{22,9}$
24	13		2^9	$b_{24,3\sim8} = b_{23,3\sim8}$, $b_{24,9} = 0$
25	3		-2^3 $(3 \sim 9)$	$b_{25,3\sim8} = 0$, $b_{24,9} = 1$
26	5			$b_{26,3\sim8} = b_{24,3\sim8}$
27	9			$b_{27,3\sim8} = b_{26,3\sim8}$, $b_{27,9} \neq b_{26,9}$
28	13			
29	3	2^3		
30	5			

The symbol $i \sim j$ for numerical difference means difference propagates from bit i to j.
The symbol $i \sim j$ for sufficient conditions means all bits from i to j.

Collisions for the LPS Expander Graph Hash Function

Jean-Pierre Tillich[1] and Gilles Zémor[2]

[1] INRIA, Équipe SECRET,
Rocquencourt, 78153 Le Chesnay, France
jean-pierre.tillich@inria.fr
[2] Institut de Mathématiques de Bordeaux,
Université Bordeaux 1,
351, cours de la Libération, 33405 Talence, France
Gilles.Zemor@math.u-bordeaux1.fr

Abstract. We analyse the hash function family based on walks in LPS Ramanujan graphs recently introduced by Charles et al. We present an algorithm for finding collisions that runs in quasi-linear time in the length of the hashed value. A concrete instance of the hash function is considered, based on a 100-digit prime. A short collision is given, together with implementation details.

1 Introduction

Given the recent profusion of efficient attacks against widely used practical hash functions, among which the MD and SHA families, there is a growing need for hash functions built upon different principles, and in particular with some degree of proven collision resistance that would come under the form: finding a collision is equivalent to solving a clearly identified mathematical problem. A promising design strategy that has been experimented with in the past and is undergoing recent developments [2,3], consists of choosing a large fixed graph that has a short and computationally efficient description, together with a natural correspondence between strings over a given alphabet and paths in the graph. The output of the hashed function is declared to be the endpoint of the path. Finding collisions is then essentially equivalent to finding cycles in the graph. If the hashed values have to be written with at least n bits, then the smallest cycle size (the girth of the graph) can be made to be at least cn for constant c, so that general purpose algorithms for finding cycles in graphs are useless because they are exponential in the cycle size.

Hash functions based on this principle were introduced in the past in [18,17,19]. The graphs are Cayley graphs over the groups G of 2×2 matrices SL_2 over finite fields of prime orders [18,19] and order a power of 2 [17]. A Cayley graph over the group G has the group elements as vertex set and there is an edge between group elements x and y if $y = xs$ where s belongs to a small, fixed, carefully chosen set $\mathcal{S}$ of group elements.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 254–269, 2008.

Finding collisions in such schemes is tantamount to factoring group elements into short non-trivial products of elements of $\mathcal{S}$. The first attempt [18] was broken in [16]. Attacks have then been mounted against [19,17] in [4,6,14]. However, a close look at these papers shows that they do not find genuine collisions. Geiselman [6] does discuss a method, but it produces collisions between input messages of astronomical size. Charnes and Pieprzyk [4] choose the field $\mathbb{F}_p$ that defines the hash function [illegible] choosing a potential collision. Similarly, Steinwandt et al. [14] choose the polynomial $P(X)$ over $\mathbb{F}_2$ that defines the hash function a posteriori. This means that if the defining parameters of the hash function (the prime p or the polynomial $P(X)$) are for example chosen to be the output of a trusted one-way function, no method is known to date for breaking these schemes. This is encouraging for SL_2 based hash functions and more generally, for hashing schemes whose collision resistance is based on the hardness of factoring in arithmetic groups.

Hashing schemes that build upon these ideas have also been proposed and discussed in [1,15]. An application to authenticating sequences and signing video images is given in [12].

In recent work [2,3], Charles et al. presented two hash function families that are also based on walking the message through a graph with arithmetic properties. The emphasis is on the expanding quality of the associated graph. Expansion is relevant to the hashing scheme because it implies the rapidly-mixing property, which means that when the input messages are sufficiently random, the output is uniformly distributed over the set of hash codes. This property stays true even when the input messages are limited to relatively small lengths. A proof of this property is clearly desirable for hashing, especially so if the hash functions are used in protocols whose security relies on the random oracle model.

In the present paper we consider the second hashing scheme of Charles et al., which is the fastest and the most likely to be considered for actual use [10]. Specifically, this scheme is based on the celebrated "Ramanujan" expander graphs of Lubotzky, Philips and Sarnak. In [3] the scheme is claimed to be an improvement over [19,17] and the underlying theoretical problem believed to be difficult. In what follows we solve the underlying problem, namely the factorisation of unity into generators of the Ramanujan Cayley graph, and provide collisions for arbitrary instances of the LPS Ramanujan hashing scheme of [3]. We exhibit an algorithm for finding collisions that runs in time quasi-linear in n, where n is the hashcode size. An actual example is discussed and implementation details are given.

The paper is organised as follows. In Section 2 we give a precise description of the hash function of [3]. In Section 3 we give an overview of the attack together with the arithmetic properties that we need. In Section 4 we provide missing details. In Section 5 we discuss a worked-out example for a Ramanujan graph based on a 100-digit prime. Dealing with 1024-bit primes is not a problem but putting the factorisation in print would be ungainly and uninformative. In Section 6 we give some comments on the attack, on possible repairs, and more generally on hashing schemes based on factoring in groups. A Maple program implementing the attack is given in the appendix.

2 The Hash Function

The cryptographic function under study, that will be denoted by H, is a particular instance of the following construction.

2.1 The General Construction

Defining parameter

A finite group G, and a set of generators $\mathcal{S}$ such that $\mathcal{S}^{-1} = \mathcal{S}$. Let $a \stackrel{\text{def}}{=} |\mathcal{S}| - 1$. Choose now a function:

$$\pi : \{0, 1, \ldots, a-1\} \times \mathcal{S} \rightarrow \mathcal{S}$$

such that for any $g \in \mathcal{S}$ the set $\pi\left(\{0, 1, \ldots, a-1\} \times \{g\}\right)$ is equal to $\mathcal{S} \setminus \{g^{-1}\}$.

Algorithm

Convert the input message to a base-a number $x_0 x_1 \ldots x_k$. Define the sequence $(g_i)_{0 \leq i \leq k}$ inductively as follows

$$g_i = \pi(x_i, g_{i-1})$$

where g_{-1} is some fixed element in $\mathcal{S}$. The hashcode of the input message is just the group element

$$H(\mathbf{x}) = g g_0 g_1 \ldots g_k$$

where g is a fixed element of G and $\mathbf{x} = (x_0, x_1, \ldots, x_k)$.

This construction is slightly more complex than the one presented in [17,19]. The idea is roughly the same: replace the symbols of the text to be hashed with group elements and multiply them together to obtain the hashed value. What is different here is the fact that the way a group element is mapped to a letter in the text depends both on the letter and on the previous associated group element. This rather involved definition is a consequence of the fact that in the generator set $\mathcal{S}$ there are pairs of generators which are inverse of each other. This implies that in order to avoid trivial collisions one should avoid having products $g_i g_{i+1}$ where $g_i = g_{i+1}^{-1}$. The way π is defined on the set $\{0, 1, \ldots, a-1\} \times \{g\}$ avoids this unwanted phenomenon.

It can be checked [3] that finding collisions for the hash function reduces to the following problem

Problem 2.1. $g_1, g_2, \ldots, g_t$ $\mathcal{S}$

$$g_1 g_2 \ldots g_t = \mathbf{1}$$
$$g_i g_{i+1} \neq \mathbf{1} \quad \forall i \in \{1, 2, \ldots, t-1\}$$

More precisely, finding a collision for H with two messages of size t' and t'' gives a solution to the previous problem for some $t \leq t' + t''$. This can checked as follows. The hashed value of the first message is the result of a product of the form $gg'_1 \dots g'_{t'}$ whereas the second one corresponds to a product $gg''_1 \dots g''_{t''}$ where the g'_i's and the g''_i's belong to $\mathcal{S}$. Both values coincide and therefore $g'_1 \dots g'_{t'} = g''_1 \dots g''_{t''}$. This implies that $g'_1 \dots g'_{t'} g''^{-1}_{t''} \dots g''^{-1}_1 = \mathbf{1}$. Conversely, consider a factorisation of the form $g_1 g_2 \dots g_t = \mathbf{1}$ with $g_1, g_2, \dots, g_t$ all in $\mathcal{S}$ and $g_i g_{i+1} \neq \mathbf{1}$ for i in $\{1, 2, \dots, t-1\}$. This implies that $g_1 g_2 \dots g_{t'} = g_t^{-1} \dots g_{t'+1}^{-1}$ for any t' in $\{1, \dots t-1\}$. If g_1^{-1} and g_t are both different from g_{-1} this yields a collision for the hash function with messages of size t' and $t - t'$ respectively. Otherwise, if $|\mathcal{S}| \geq 4$ there exists g' in $\mathcal{S}$ such that $g' \in \mathcal{S} \backslash \{g_{-1}^{-1}, g_1^{-1}, g_t\}$. Observe now that $gg'g_1g_2 \dots g_{t'} = gg'g_t^{-1} \dots g_{t'+1}^{-1}$ and that the first term corresponds to the hashed value of a message of size $t'+1$ whereas the second term corresponds to the hashed value of a message of size $t - t' + 1$.

As explained in the introduction, the [illegible] associated to G and $\mathcal{S}$ has vertex set G and there is an edge between x and y if and only if y is equal to $x.g$ for some g in $\mathcal{S}$. Calculating the hashcode $gg_1g_2 \dots g_t$ of a t-symbol long input message amounts to taking a non-backtracking walk in the graph by starting at vertex g and performing the following steps

$$g \xrightarrow{g_1} gg_1 \xrightarrow{g_2} gg_1g_2 \rightarrow \cdots \xrightarrow{g_t} gg_1g_2 \dots g_t.$$

This walk is non-backtracking since we do not allow products of the form $g_i g_{i+1}$ that would be equal to the identity.

The particular Cayley graph chosen by the authors of [3] (whose defining group G and generator set $\mathcal{S}$ are presented in Subsection 2.2 below) is the celebrated Ramanujan graph construction of Lubotzky, Phillips, Sarnak (LPS) [11]. It has two properties relevant to hashing.

First, the graph is a good expander (see [3], [13] for details and [9] for a modern survey on expander graphs). This implies among other things that a random walk in this graph is close to the uniform distribution when the length of the walk is some constant times the logarithm of the number of vertices. From this property it is readily seen that the distribution of the hashed values is close to the uniform distribution as soon as the text size is some constant times the hashcode size. In the Ramanujan graph case, the size of the constant is quite small (slightly above 2 will do the job here).

Second, the LPS graph has no small cycles. This ensures that solutions to Problem 2.1 are large enough to make exhaustive search hopeless.

2.2 The Particular Choice of [2,3]

The authors of [2,3] choose for G the group $\mathrm{PSL}_2(\mathbb{F}_p)$: recall that $\mathrm{SL}_2(\mathbb{F}_p)$ is the group of 2×2 matrices of determinant 1 with entries in $\mathbb{F}_p$ and $\mathrm{PSL}_2(\mathbb{F}_p)$ is obtained from $\mathrm{SL}_2(\mathbb{F}_p)$ by taking the quotient by its centre, that is $\{1, -1\}$. This amounts to identifying matrix A with $-A$. The group $\mathrm{PSL}_2(\mathbb{F}_p)$ is of size $p(p^2-1)/2$. The prime p is chosen congruent to 1 modulo 4. The size of the generator set $\mathcal{S}$ will be equal to $\ell+1$ where ℓ is a small prime congruent to 1 modulo 4 and which is also a quadratic residue $\pmod{p}$. The generators are obtained from the $\ell+1$ integer solutions (a,b,c,d) of the Diophantine equation

$$\begin{cases} a^2+b^2+c^2+d^2=\ell \\ \qquad\qquad a>0, \quad a\equiv 1 \pmod{2} \\ \qquad\qquad b\equiv c\equiv d\equiv 0 \pmod{2} \end{cases} \tag{1}$$

For a proof that the number of solutions to (1) is indeed $\ell+1$ see for example [5, Ch. 2]. To each such (a,b,c,d) we associate the 2×2 matrix with entries in the ring $\mathbb{Z}[i]$ of Gaussian integers

$$M(a,b,c,d)=\begin{pmatrix} a+ib & c+id \\ -c+id & a-ib \end{pmatrix}. \tag{2}$$

We then map the entries of $M(a,b,c,d)$ to elements of $\mathbb{F}_p$ by applying the ring homomorphism

$$\begin{aligned} \phi\ &: \mathbb{Z}[i]\rightarrow \mathbb{F}_p \\ a+ib &\mapsto a+\iota b \end{aligned} \tag{3}$$

where ι is a square root of -1 modulo p (which lies in $\mathbb{F}_p$ since $p\equiv 1 \pmod{4}$). After applying ϕ we denote the resulting matrices by $\widetilde{M}(a,b,c,d)$. Note that

Fact 2.2. [illegible] $\widetilde{M}(a,b,c,d)$ [illegible] $\ell \pmod{p}$

We now view the matrices $\widetilde{M}(a,b,c,d)$ as elements of $\mathrm{PGL}_2(\mathbb{F}_p)$. Recall that this is the group of 2×2 invertible matrices with entries in $\mathbb{F}_p$ obtained after identifying pairs of matrices M and N whenever there exists $\lambda\in\mathbb{F}_p$ such that $M=\lambda N \pmod{p}$.

The set of generators $\mathcal{S}$ is now declared to be the set of matrices $\widetilde{M}(a,b,c,d)$ in $\mathrm{PGL}_2(\mathbb{F}_p)$.

Note that $\mathcal{S}^{-1}=\mathcal{S}$. This comes from the fact that in $\mathrm{PGL}_2(\mathbb{F}_p)$ we have

$$\begin{aligned} \widetilde{M}(a,b,c,d)\widetilde{M}(a,-b,-c,-d) &= \begin{pmatrix} a+\iota b & c+\iota d \\ -c+\iota d & a-\iota b \end{pmatrix}\begin{pmatrix} a-\iota b & -c-\iota d \\ c-\iota d & a+\iota b \end{pmatrix} \\ &= \begin{pmatrix} a^2+b^2+c^2+d^2 & 0 \\ 0 & a^2+b^2+c^2+d^2 \end{pmatrix} \\ &\equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

Finally, it should also be noted that $\mathcal{S}$ does not generate the whole group $\mathrm{PGL}_2(\mathbb{F}_p)$. This comes from the fact that all the matrices in $\mathcal{S}$ have determinant ℓ which is a square modulo p. Therefore only matrices with determinant that are quadratic residues $\pmod{p}$ are generated. It can be checked that the generated subgroup G is isomorphic to $\mathrm{PSL}_2(\mathbb{F}_p)$ (see [11,13]).

The Cayley graph associated to G and $\mathcal{S}$ is denoted by $X_{\ell,p}$ and is the LPS Ramanujan graph mentioned above. Apart from its expansion properties, the graph $X_{\ell,p}$ has a girth (smallest cycle length) at least $2\log_\ell p$ (see [11,13]). Practical sizes of the parameters would be a prime p of several hundred bits (say 1024) and a small prime ℓ (say 5). This would mean that the smallest solution to Problem 2.1 must involve at least 882 generators.

3 An Outline of the Attack

Factoring in $\mathrm{PGL}_2(\mathbb{F}_p)$ directly seems difficult. Our strategy will be to first lift matrices of $\mathrm{PGL}_2(\mathbb{F}_p)$ into a set of matrices with entries in $\mathbb{Z}[i]$, and then factor into a product of lifted generators of $\mathcal{S}$, namely the matrices of (2). The relevant set of matrices is

$$\Omega = \left\{ \begin{pmatrix} a+ib & c+id \\ -c+id & a-ib \end{pmatrix} \middle| (a,b,c,d) \in E_w \text{ for some integer } w > 0 \right\}$$

where E_w is the set of 4-tuples $(a,b,c,d) \in \mathbb{Z}^4$ such that

$$\begin{cases} a^2+b^2+c^2+d^2 = \ell^w \\ \qquad a > 0, \quad a \equiv 1 \pmod 2 \\ \qquad b \equiv c \equiv d \equiv 0 \pmod 2. \end{cases} \tag{4}$$

Consider now the set Σ of $\ell+1$ matrices $M(a,b,c,d)$ with $\mathbb{Z}[i]$ entries defined in (2). In other words, Σ is the subset of Ω corresponding to 4-tuples (a,b,c,d) in E_1, and it is also the lifted version of the set of generators $\mathcal{S}$. It turns out that the set Ω coincides, up to multiplication by ± 1 and by powers of ℓ, with products of elements of Σ. Precisely, we have the following lemma which is a reformulation of Corollary 3.2 of [11]: for more details, see also [13, Lemma 2.5.4] or [5, Corollary 2.6.14].

Lemma 3.1. [illegible] M [illegible] Ω [illegible]

$$M = \pm \ell^r M_1 M_2 \dots M_e$$

[illegible] $\log_\ell(\det M) = e + 2r$ [illegible] M_i [illegible] Σ [illegible] $M_i M_{i+1} \neq \ell\,\mathbf{1}$ [illegible] $i \in \{1, \dots, e-1\}$

The attack now proceeds along the following lines.

Step 1 (lifting the identity in Ω): The aim of this step is to find a matrix M in Ω which is not of the form $\ell^r\mathbf{1}$ and such that if we replace the complex entries by their "corresponding" values in $\mathbb{F}_p$ (i.e. apply the mapping ϕ (3)) then we obtain a matrix $\widetilde{M}$ of the form $\lambda\mathbf{1}$. This amounts to finding a, b, c, d in $\mathbb{Z}$ such that

$$\begin{cases} a^2+b^2+c^2+d^2 = \ell^w \\ \qquad a > 0, \quad a \equiv 1 \pmod 2 \\ \qquad b \equiv c \equiv d \equiv 0 \pmod{2p} \\ \quad b^2+c^2+d^2 \neq 0 \end{cases} \tag{5}$$

for some positive integer w.

Step 2 (factorisation step): Find the factorisation of M promised by Corollary 3.1:

$$M = \pm \ell^r M_1 M_2 \dots M_e.$$

What really makes this task simple is the fact that the factorisation is unique. Proceed as follows. We first find the greatest integer r such that ℓ^r divides the 4 entries of M. Let M' be such that $M = \ell^r M'$. We denote by $G_1, \dots, G_{\ell+1}$ the $\ell + 1$ elements forming the lifted generator set Σ.

We start by finding the rightmost element in the factorisation of M' by computing all products of the form $\pm M' G_i^{-1}$. Necessarily one of these products will be in Ω: it will correspond to the factorisation of M' where we have dropped the last element of the factorisation. It is impossible that there is more than one of these products which lies in Ω: by using the fact that every element of Ω can be factored into elements of Σ we would obtain at least two different factorisations for M'. This would contradict the unique factorisation property. Therefore, the unique G_i for which $\pm M' G_i^{-1}$ belongs to Ω is the last element of the factorisation of M'. Notice that checking whether a product $\pm M' G_i^{-1}$ is in Ω or not is computationally easy given the definition of Ω. We continue this process and it has to stop after $\log_\ell(\det M) - 2r$ steps because at each iteration the determinant of the left part of the factorisation gets divided by ℓ and because M' is of determinant ℓ^{w-2r}. The complexity of this step is obviously proportional to the length of the factorisation, i.e. at most w. We will see below that we can choose w to be approximately $2 \log_\ell p$, so that the complexity is not more than $O(\log p)$.

Step 3 (final step): The point is that the matrix $\widetilde{M}$ with entries in $\mathbb{F}_p$ reduces to the identity in $\mathrm{PGL}_2(\mathbb{F}_p)$ and can be factored in this group by using the $\ell + 1$ generators of $\mathcal{S}$ as follows:

$$\mathbf{1} \equiv \widetilde{M} \equiv \widetilde{M_1}\widetilde{M_2} \dots \widetilde{M_e}$$

where $\widetilde{M_i}$ is the application of the aforementioned homomorphism ϕ to each entry of M_i (meaning $\widetilde{M_i}$ belongs to $\mathcal{S}$). This solves Problem 2.1.

4 Solving Step 1

Solving Equation (5) seems to be easier when w is even. So let us arbitrarily set $w = 2k$ and let us write $b = 2px, c = 2py, d = 2pz$. We are looking for integer solutions (a, x, y, z) to the equation

$$a^2 + 4p^2(x^2 + y^2 + z^2) = \ell^{2k}.$$

This implies that

$$(\ell^k - a)(\ell^k + a) = 4p^2(x^2 + y^2 + z^2)$$

Let us choose $a = \ell^k - 2mp^2$ for some integer m. In this case, $(\ell^k - a)(\ell^k + a) = 2mp^2(2\ell^k - 2mp^2)$. Thus x, y, z should satisfy the equation

$$x^2 + y^2 + z^2 = m(\ell^k - mp^2) \tag{6}$$

Let us now specify how m and k are chosen. a should be positive and k as small as possible in order to minimise the length of the factorisation of the identity obtained at the end. We choose k to be the smallest integer such that $\ell^k - 4p^2 > 0$. We may then either choose $m = 1$ or $m = 2$ in order to keep a positive. We claim now that for either $m = 1$ or $m = 2$ the number $m(\ell^k - mp^2)$ is a sum of three squares. Let us recall Legendre's theorem (see [7]) which asserts that all integers are sums of 3 squares with the exception of the integers of the form $4^s(8t + 7)$ where s and t are integers. Assume that $m = 1$ does not work. In other words, $\ell^k - p^2$ is not a sum of three squares. This means that there exists s and t which are non-negative integers such that $\ell^k - p^2 = 4^s(8t + 7)$. Note that s has to be positive in this case. Observe now that $2(\ell^k - 2p^2) = 4^s(16t + 14) - 2p^2$. This number is neither a multiple of 4 nor odd. Therefore it can not be of the form $4^u(8v + 7)$. This shows that $m = 2$ is suitable for our purpose.

It remains now to find x, y, z which satisfy Equation (6). One way of achieving this goal is to subtract from $m(\ell^k - mp^2)$ a random x^2 and to hope that the result N is a sum of 2 squares. In this case there is a simple and efficient algorithm relying on Euclid's algorithm for finding y and z explicitly such that $y^2 + z^2 = N$. Fermat's theorem (see [7]) on sums of two squares says that a number is expressible as a sum of two squares if and only if its prime factors congruent to 3 modulo 4 occur with an even exponent. Our approach is to try to find values of x for which N is of the form $2^s p'$ where p' is a prime congruent to 1 modulo 4. When $m = 1$ for instance, we choose even values of x and since $\ell^k - p^2 \equiv 0 \mod 4$ we check whether $(\ell^k - p^2 - x^2)/4$ is a prime congruent to 1 modulo 4. This happens roughly with probability of order $O(1/\ln(\ell^k - p^2))$.

It remains to explain how we find y and z such that

$$y^2 + z^2 = N. \tag{7}$$

This is classical and can be done by using continued fraction expansion. We give the details for the sake of self-sufficiency and to explain implementation details. Let us recall that the convergents $\frac{p_n}{q_n}$ of the continued fraction expansion of a real number x are obtained inductively from the formulas

$$\begin{aligned}(p_{-1}, q_{-1}) &= (0, 1)\\ (p_0, q_0) &= (1, 0)\end{aligned}$$

and for all nonnegative values of n for which $q_n x - p_n \neq 0$

$$\begin{aligned}a_n &= \left[-\frac{q_{n-1}x - p_{n-1}}{q_n x - p_n}\right]\\ p_{n+1} &= a_n p_n + p_{n-1}\\ q_{n+1} &= a_n q_n + q_{n-1}\end{aligned}$$

where [ ] denotes the integer part.

The sequence $(q_n)_{n \geq 0}$ is strictly increasing and the $\frac{p_n}{q_n}$ are very good rational approximations of x. They satisfy:

Proposition 4.1.

$$\left|x - \frac{p_n}{q_n}\right| \leq \frac{1}{q_n q_{n+1}}.$$

From Fermat's theorem and the identity

$$(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (bc + ad)^2$$

we know that in order to find integer solutions of Equation (7) we just need to solve this kind of equation when N is a prime congruent to 1 modulo 4. In this case, -1 is a quadratic residue modulo N. This fact is used as follows

Proposition 4.2. N 1 4 R -1 N $\xi \stackrel{def}{=} \frac{R}{N}$ $\frac{p_i}{q_i}$ ξ n $q_n < \sqrt{N} < q_{n+1}$

$$q_n^2 + (q_n R - p_n N)^2 = N.$$

First of all it should be noticed that such an n exists: the sequence of the q_i's is increasing and is defined up to the term q_j such that $q_j = N$. It follows from Proposition 4.1 that $\left|\frac{R}{N} - \frac{p_n}{q_n}\right| < \frac{1}{q_n q_{n+1}}$. Hence $\left|q_n \frac{R}{N} - p_n\right| < \frac{1}{q_{n+1}} < \frac{1}{\sqrt{N}}$ (because $q_{n+1} > \sqrt{N}$). This implies that $|q_n R - p_n N| < \sqrt{N}$. We also have $q_n < \sqrt{N}$. Putting both inequalities together we obtain $q_n^2 + (q_n R - p_n N)^2 < 2N$. Let us notice now that

$$\begin{aligned} q_n^2 + (q_n R - p_n N)^2 &\equiv q_n^2 + q_n^2 R^2 \pmod{N} \\ &\equiv 0 \pmod{N} \end{aligned}$$

Therefore, we necessarily have that $q_n^2 + (q_n R - p_n N)^2 = N$.

The exact complexity of this step is unclear, this is due to the problem of estimating the time complexity for finding an N whose prime factors congruent to 3 modulo 4 all occur with an even exponent. We can upper bound this quantity by the complexity for finding an N of the form $2^s q$, where q is a prime congruent to 1 modulo 4 and heuristic arguments based on the density of primes congruent to 1 modulo 4 indicate that the number of x's which have to be tried in order to find a proper N will be of order $O(\log p)$. The complexity for performing the continued fraction expansion is also of order $O(\log p)$. Therefore the total complexity of this step should be extremely low and will be of order $O(\log p)$. This has been confirmed experimentally (see Section 5).

5 An Example of an Attack

In this example we take $p = 10^{100} + 949$ which is the first prime $p > 10^{100}$ such that $p = 1 \mod 4$. Computations were done with a Maple program given in the appendix.

Next, consider the hash function corresponding to the graph $X_{5,p}$, i.e. we set $\ell = 5$, the first possible case since we must have $\ell = 1 \bmod 4$. It turns out that 5 is a quadratic residue for this choice of p. The 6 generators of $X_{5,p}$ are given by the matrices with $\mathbb{Z}[i]$ entries,

$$G_1 = \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \quad G_2 = \begin{pmatrix} 1+2i & 0 \\ 0 & 1-2i \end{pmatrix} \quad G_3 = \begin{pmatrix} 1 & 2i \\ 2i & 1 \end{pmatrix}$$
$$G_4 = \begin{pmatrix} 1 & -2i \\ -2i & 1 \end{pmatrix} \quad G_5 = \begin{pmatrix} 1-2i & 0 \\ 0 & 1+2i \end{pmatrix} \quad G_6 = \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix}$$

or by their images $\widetilde{G_j}$ in $\mathrm{PGL}_2(\mathbb{F}_p)$.

Step 1. We first look for a, b, c, d satisfying (5). We choose k to be the first integer larger than $\log_5(2p^2)$ to make the right hand side of (6) positive. We obtain $k = 287$. We then compute $5^k - p^2$ which is of the form $4u$ with u odd. As was quite likely, we have $u \neq 7 \mod 8$, which means that u can be expressed as a sum of three squares. Furthermore, it turns out that we have $u = 1 \mod 4$, so we try subtracting from u squares of the form $4v^2$ and test $u - 4v^2$ for primality. When we meet a prime, it will necessarily be congruent to 1 modulo 4, so that we will be able to express it as a sum of two squares. The first v such that $N = u - 4v^2$ is prime is $v = 1431$.

We then proceed to express N as a sum of two squares. We first find a square root R of -1 modulo N. We arbitrarily choose the root whose representation in $\{1, \ldots, N-1\}$ is largest. We then expand R/N into a continued fraction and compute the largest n such that p_n/q_n is the nth convergent of the continued fraction expansion and $q_n < \sqrt{N}$. In this particular case we find $n = 192$. We then set

$$\begin{aligned} x &= 2q_n \\ y &= 2(p_n N - q_n R) \\ z &= 4 \times 1431 = 5724 \end{aligned}$$

and obtain $5^k - p^2 = 4u = x^2 + y^2 + z^2$. We then set

$$\begin{aligned} a &= \sqrt{5^{2k} - 16up^2} \text{ which is an integer} \\ b &= 2px \\ c &= 2py \\ d &= 2pz \end{aligned}$$

Step 2. We now factor in Ω the matrix

$$M = \begin{pmatrix} a+ib & c+id \\ -c+id & a-ib \end{pmatrix}$$

into a product of G_j's. We know that there is a unique way to do this, and that the length of the factorisation is $2k = 574$, i.e.

$$M = \pm M_1 \ldots M_{2k}$$

where $M_j = G_{\sigma(j)}$ with $\sigma(j) \in \{1, 2, \ldots, 6\}$. To compute σ, we first multiply on the right M by all six matrices $G_1 \ldots G_6$, and test whether all entries are multiples of 5. When this happens we have found $G_{\sigma(574)}$. We compute $M' = MG^{-1}_{\sigma(574)} = \frac{1}{5}MG_{7-\sigma(574)}$ and proceed recursively, testing $M'G_j$ at most six times to obtain $\sigma(573)$, and so on. After 574 iterations were are left either with the identity matrix $\mathbf{1}$ or with $-\mathbf{1}$. The first 24 values $\sigma(1), \sigma(2), \ldots, \sigma(24)$ of σ are

$$2, 4, 2, 3, 3, 3, 3, 1, 1, 4, 1, 5, 5, 5, 5, 1, 5, 1, 1, 1, 4, 1, 4, 6,$$

and the remaining 550 values are given by the array in figure 1, each row giving the next 25 values. We have exhibited the factorisation of unity

6, 2, 1, 2, 3, 2, 2, 3, 1, 1, 1, 3, 1, 2, 2, 1, 2, 6, 6, 6, 3, 1, 5, 4, 1,
4, 5, 1, 1, 3, 2, 3, 6, 5, 5, 5, 3, 3, 5, 5, 6, 2, 4, 1, 1, 5, 3, 1, 5, 1,
2, 1, 2, 1, 5, 6, 4, 1, 4, 4, 4, 6, 5, 1, 5, 3, 1, 2, 2, 4, 1, 4, 5, 4, 1,
3, 6, 3, 3, 1, 4, 6, 3, 5, 5, 6, 4, 6, 3, 3, 1, 2, 3, 3, 2, 4, 5, 3, 5, 4,
5, 4, 2, 2, 2, 4, 6, 4, 1, 1, 4, 2, 3, 1, 4, 5, 4, 6, 5, 5, 3, 1, 4, 5, 6,
2, 1, 2, 6, 2, 1, 3, 3, 2, 6, 6, 5, 1, 5, 3, 1, 5, 1, 5, 1, 2, 6, 3, 3, 1,
1, 1, 4, 2, 1, 1, 3, 5, 6, 4, 6, 2, 6, 6, 3, 6, 2, 6, 6, 6, 2, 4, 1, 2, 6,
5, 3, 1, 4, 1, 2, 6, 4, 4, 2, 4, 4, 2, 1, 2, 4, 4, 1, 2, 2, 2, 2, 6, 3, 2,
1, 2, 4, 2, 6, 2, 2, 4, 4, 1, 1, 1, 1, 2, 6, 2, 4, 5, 3, 2, 4, 1, 1, 1, 4,
2, 2, 1, 1, 1, 3, 1, 5, 6, 2, 4, 5, 5, 1, 4, 1, 3, 2, 6, 6, 4, 6, 4, 6, 4,
6, 3, 1, 1, 2, 6, 3, 2, 6, 6, 6, 3, 1, 2, 4, 2, 3, 3, 3, 3, 1, 1, 4, 1, 5,
5, 5, 5, 1, 5, 1, 1, 1, 4, 1, 4, 6, 6, 2, 1, 2, 3, 2, 2, 3, 1, 1, 1, 3, 1,
2, 2, 1, 2, 6, 6, 6, 3, 1, 5, 4, 1, 4, 5, 1, 1, 3, 2, 3, 6, 5, 5, 5, 3, 3,
5, 5, 6, 2, 4, 1, 1, 5, 3, 1, 5, 1, 2, 1, 2, 1, 5, 6, 4, 1, 4, 4, 4, 6, 5,
1, 5, 3, 1, 2, 2, 4, 1, 4, 5, 4, 1, 3, 6, 3, 3, 1, 4, 6, 3, 5, 5, 6, 4, 6,
3, 3, 1, 2, 3, 3, 2, 4, 5, 3, 5, 4, 5, 4, 2, 2, 2, 4, 6, 4, 1, 1, 4, 2, 3,
1, 4, 5, 4, 6, 5, 5, 3, 1, 4, 5, 6, 2, 1, 2, 6, 2, 1, 3, 3, 2, 6, 6, 5, 1,
5, 3, 1, 5, 1, 5, 1, 2, 6, 3, 3, 1, 1, 1, 4, 2, 1, 1, 3, 5, 6, 4, 6, 2, 6,
6, 3, 6, 2, 6, 6, 6, 2, 4, 1, 2, 6, 5, 3, 1, 4, 1, 2, 6, 4, 4, 2, 4, 4, 2,
1, 2, 4, 4, 1, 2, 2, 2, 2, 6, 3, 2, 1, 2, 4, 2, 6, 2, 2, 4, 4, 1, 1, 1, 1,
2, 6, 2, 4, 5, 3, 2, 4, 1, 1, 1, 4, 2, 2, 1, 1, 1, 3, 1, 5, 6, 2, 4, 5, 5,
1, 4, 1, 3, 2, 6, 6, 4, 6, 4, 6, 4, 6, 3, 1, 1, 2, 6, 3, 2, 6, 6, 6, 3, 1.

Fig. 1. The remaining 550 values $\sigma(25), \ldots, \sigma(574)$

$$\mathbf{1} = \widetilde{G}_{\sigma(1)} \widetilde{G}_{\sigma(2)} \ldots \widetilde{G}_{\sigma(574)}$$

in $\mathrm{PGL}_2(\mathbb{F}_p)$. This can easily be checked with the program given in the appendix. Running time is counted in seconds rather than minutes, and stays that way if p is replaced by a 1024-bit prime.

6 Comments

The attack presented here is somewhat reminiscent of the "density attack" (to the terminology of [17]) that was used in [16] to break the hashing scheme first proposed in [18]. In that attack the group unit element is first lifted into a "dense" subset of $\mathrm{SL}_2(\mathbb{Z})$ and then a factorisation algorithm is applied in $\mathrm{SL}_2(\mathbb{Z})$.

Can the "Ramanujan" hash function family be fixed so as to make the present attack unfeasible ? Well, there are several natural solutions to address this problem. One idea is to change the set of generators from $\mathcal{S}$ to $\mathcal{S}^2$ (we square the elements of $\mathcal{S}$) for example. That way the present attack will succeed only if one manages to lift the identity element of G onto a matrix of Ω that has a very special (and rare) factorisation into elements of $\mathcal{S}$. A similar idea is to reduce the set $\mathcal{S}$ by throwing away some generators. In this case though, one must be careful to ensure that the modified set of generators generates the same subgroup of $\mathrm{PGL}_2(\mathbb{F}_p)$ as the original generator set. It is also unclear what will happen to the expansion properties when modifying the hash function in this way, and more study is required to come up with suitable choices.

The very property that makes the graphs $X_{\ell,p}$ Ramanujan gave us a tool for mounting an attack, so resorting to these highly structured Cayley graphs may not be the best idea if one is to base a hash function on factoring in arithmetic groups like SL_2 (or PSL_2 or PGL_2). However, for hashing purposes, lesser guaranteed expansion properties may be sufficient. A promising result in that direction is the recent paper of Helfgott [8] which shows that, for any generating set $\mathcal{S}$ of $G = \mathrm{SL}_2(\mathbb{F}_p)$, any element of G can be expressed as a product of elements of $\mathcal{S}$ of length not more than $O(\log^c p)$. This falls somewhat short of the rapidly-mixing property, but it does guarantee that if any such Cayley graph is used as the basis of a hashing scheme, then over a set of relatively small-length input messages, the corresponding set of hashed values ranges over the whole group G.

References

1. Abdukhalikov, K.S., Kim, C.: On the security of the hashing scheme based on SL2. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 93–102. Springer, Heidelberg (1998)
2. Charles, D.X., Goren, E.Z., Lauter, K.E.: Cryptographic hash functions from expander graphs. In: Second NIST cryptographic hash workshop, Santa Barbara, USA (August 2006)
3. Charles, D.X., Goren, E.Z., Lauter, K.E.: Cryptographic hash functions from expander graphs. Journal of Cryptology, to appear in print, published online (September 15, 2007) http://www.springerlink.com/content/cv4r187833614475/
4. Charnes, C., Pieprzyk, J.: Attacking the SL2 hashing scheme. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) ASIACRYPT 1994. LNCS, vol. 917, pp. 322–330. Springer, Heidelberg (1995)
5. Davidoff, G., Sarnak, P., Valette, A.: Elementary Number Theory, Group Theory, and Ramanujan Graphs. Cambridge University Press, Cambridge (2003)
6. Geiselmann, W.: A note on the hash function of Tillich and Zémor. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 51–52. Springer, Heidelberg (1996)
7. Grosswald, E.: Representation of integers as sum of squares. Springer, Heidelberg (1985)

8. Helfgott, H.A.: Growth and generation in SL2(Z/pZ), to appear in Annals of Math. (2005), http://arxiv.org/abs/math/0509024
9. Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bull. AMS 43, 439–561 (2006)
10. Lauter, K.E., Charles, D.X., Goren, E.Z.: Hash function constructions from expander graphs, United States Patent 20070098150 (May 2007), http://www.freepatentsonline.com/20070098150.html
11. Lubotzky, A., Phillips, R., Sarnak, P.: Ramanujan graphs. Combinatorica 8(3), 261–277 (1988)
12. Quisquater, J., Joye, M.: Authentication of sequences with the SL2 hash function: Application to video sequences. Journal of Computer Security 5(3), 213–223 (1997)
13. Sarnak, P.: Some applications of modular forms. Cambridge U. Press, Cambridge (1990)
14. Steinwandt, R., Grassl, M., Geiselmann, W., Beth, T.: Weaknesses in the SL2 hashing scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 287–299. Springer, Heidelberg (2000)
15. Shpilrain, V.: Hashing with polynomials. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 22–28. Springer, Heidelberg (2006)
16. Tillich, J.-P., Zémor, G.: Group-theoretic hash functions. In: Cohen, G., Lobstein, A., Zémor, G., Litsyn, S.N. (eds.) Algebraic Coding 1993. LNCS, vol. 781, pp. 90–110. Springer, Heidelberg (1994)
17. Tillich, J.-P., Zémor, G.: Hashing with SL_2. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 40–49. Springer, Heidelberg (1994)
18. Zémor, G.: Hash functions and graphs with large girth. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, Springer, Heidelberg (1991)
19. Zémor, G.: Hash functions and Cayley graphs. Designs, Codes and Cryptography 4, 381–394 (1994)

A Appendix

Setting up the parameters of the hash function.

```
> p:= nextprime(10^100):
> p mod 4;
                              3
> p := nextprime(p):
> p mod 4;
                              1
> with(numtheory):
> legendre(5,p);
                              1
```

Step 1.

```
> evalf(log(2*p^2)/log(5));
                         286.5659883
> k:=287;
                           k := 287
> Z:=5^287-p^2:
```

```
> Z mod 4;
                                   0
> Z mod 8;
                                   4
> u:=Z/4:
> u mod 4;
                                   1
```

Finding a square that substracted to u yields a prime $N = 1 \mod 4$.

```
> for v to 10000 do if isprime(u-4*v^2)=true then print(v): fi: od:
                                  1431
                                  1794
                                  4434
                                  5610
                                  6555
                                  6666
                                  8484
                                  9405
> N:= u-4*1431^2:
> N mod 4;
                                   1
```

Expressing the prime N as a sum of two squares.

```
> R:= Roots(X^2+1) mod N:
> T:=R[1][1]:
> cf := cfrac(T/N):
> n:=0:
> while nthdenom(cf,n+1)<evalf(sqrt(N)) do n:=n+1:  od: print(n);
                                  192
> Q:=nthdenom(cf,192): P := nthnumer(cf,192):
> x:=2*Q:
> y:=2*(Q*T-P*N):
> z:=4*1431;
                               z := 5724
```

Checking that we have found three squares that sum to $4u$.

```
> x^2+y^2+z^2 -4*u;
                                   0
```

Defining the matrix M.

```
> 5^(2*k)-4*p^2*(x^2+y^2+z^2):
> a:= sqrt(%):
> b:=2*p*x:
> c:=2*p*y:
```

```
> d:=2*p*z:
> with(LinearAlgebra):
> M := Matrix(2,2):
> M[1,1]:=a+I*b:
> M[2,2]:=a-I*b:
> M[1,2]:=c+I*d:
> M[2,1]:=-c+I*d:
```

Checking the length of the factorisation of M that should be $2k$.

```
> Determinant(M):
> eval(log(%)/log(5));
                                574
```

Step 2.
Define the set of generators $G_1, \dots, G_6$.

```
> G[1] := Matrix(2,2):
> G[1][1,1]:=1: G[1][1,2]:=2: G[1][2,1]:=-2: G[1][2,2]:=1:
> G[1];
                              [ 1  2]
                              [     ]
                              [-2  1]
> G[6] := Matrix(2,2):
> G[6][1,1]:=1: G[6][2,2]:=1: G[6][1,2]:=-2: G[6][2,1]:=2:
> G[2] := Matrix(2,2):
> G[2][1,1]:=1+2*I: G[2][2,2]:=1-2*I: G[2][1,2]:=0: G[2][2,1]:=0:
> G[5]:=Matrix(2,2):
> G[5][1,1]:=1-2*I: G[5][2,2]:=1+2*I: G[5][1,2]:=0: G[5][2,1]:=0:
> G[3]:=Matrix(2,2):
> G[3][1,1]:=1: G[3][2,2]:=1: G[3][1,2]:=2*I: G[3][2,1]:=2*I:
> G[4]:=Matrix(2,2):
> G[4][1,1]:=1: G[4][2,2]:=1: G[4][1,2]:=-2*I: G[4][2,1]:=-2*I:
```

The procedure that factors M into a product of G_i's.

```
> fact := proc(m,MM)
> local L,H,i,j,X:
> L:=[]: H[1]:=MM:
> for i to m do
> for j to 6 do
> X := Multiply(H[i],G[j]):
> if (X[1,1] mod 5)=0 and (X[1,2] mod 5)=0 and (X[2,1] mod 5)=0
> and (X[2,2] mod 5)=0 then
> H[i+1]:=X/5: L := [7-j,op(L)]: fi: od:
> od:
> print(H[m+1]); return(L);
> end proc:
```

The actual factorisation for this particular example.

```
> F:=fact(574,M);
                              [-1  0]
                              [     ]
                              [0  -1]

F := [2, 4, 2, 3, 3, 3, 3, 1, 1, 4, 1, 5, 5, 5, 5, 1, 5, 1, 1, 1, 4, 1, 4, 6,
  6, 2, 1, 2, 3, 2, 2, 3, 1, 1, 1, 3, 1, 2, 2, 1, 2, 6, 6, 6, 3, 1, 5, 4, 1,
  4, 5, 1, 1, 3, 2, 3, 6, 5, 5, 5, 3, 3, 5, 5, 6, 2, 4, 1, 1, 5, 3, 1, 5, 1,
  2, 1, 2, 1, 5, 6, 4, 1, 4, 4, 4, 6, 5, 1, 5, 3, 1, 2, 2, 4, 1, 4, 5, 4, 1,
  3, 6, 3, 3, 1, 4, 6, 3, 5, 5, 6, 4, 6, 3, 3, 1, 2, 3, 3, 2, 4, 5, 3, 5, 4,
  5, 4, 2, 2, 2, 4, 6, 4, 1, 1, 4, 2, 3, 1, 4, 5, 4, 6, 5, 5, 3, 1, 4, 5, 6,
  2, 1, 2, 6, 2, 1, 3, 3, 2, 6, 6, 5, 1, 5, 3, 1, 5, 1, 5, 1, 2, 6, 3, 3, 1,
  1, 1, 4, 2, 1, 1, 3, 5, 6, 4, 6, 2, 6, 6, 3, 6, 2, 6, 6, 6, 2, 4, 1, 2, 6,
  5, 3, 1, 4, 1, 2, 6, 4, 4, 2, 4, 4, 2, 1, 2, 4, 4, 1, 2, 2, 2, 2, 6, 3, 2,
  1, 2, 4, 2, 6, 2, 2, 4, 4, 1, 1, 1, 1, 2, 6, 2, 4, 5, 3, 2, 4, 1, 1, 1, 4,
  2, 2, 1, 1, 1, 3, 1, 5, 6, 2, 4, 5, 5, 1, 4, 1, 3, 2, 6, 6, 4, 6, 4, 6, 4,
  6, 3, 1, 1, 2, 6, 3, 2, 6, 6, 6, 3, 1, 2, 4, 2, 3, 3, 3, 3, 1, 1, 4, 1, 5,
  5, 5, 5, 1, 5, 1, 1, 1, 4, 1, 4, 6, 6, 2, 1, 2, 3, 2, 2, 3, 1, 1, 1, 3, 1,
  2, 2, 1, 2, 6, 6, 6, 3, 1, 5, 4, 1, 4, 5, 1, 1, 3, 2, 3, 6, 5, 5, 5, 3, 3,
  5, 5, 6, 2, 4, 1, 1, 5, 3, 1, 5, 1, 2, 1, 2, 1, 5, 6, 4, 1, 4, 4, 4, 6, 5,
  1, 5, 3, 1, 2, 2, 4, 1, 4, 5, 4, 1, 3, 6, 3, 3, 1, 4, 6, 3, 5, 5, 6, 4, 6,
  3, 3, 1, 2, 3, 3, 2, 4, 5, 3, 5, 4, 5, 4, 2, 2, 2, 4, 6, 4, 1, 1, 4, 2, 3,
  1, 4, 5, 4, 6, 5, 5, 3, 1, 4, 5, 6, 2, 1, 2, 6, 2, 1, 3, 3, 2, 6, 6, 5, 1,
  5, 3, 1, 5, 1, 5, 1, 2, 6, 3, 3, 1, 1, 1, 4, 2, 1, 1, 3, 5, 6, 4, 6, 2, 6,
  6, 3, 6, 2, 6, 6, 6, 2, 4, 1, 2, 6, 5, 3, 1, 4, 1, 2, 6, 4, 4, 2, 4, 4, 2,
  1, 2, 4, 4, 1, 2, 2, 2, 2, 6, 3, 2, 1, 2, 4, 2, 6, 2, 2, 4, 4, 1, 1, 1, 1,
  2, 6, 2, 4, 5, 3, 2, 4, 1, 1, 1, 4, 2, 2, 1, 1, 1, 3, 1, 5, 6, 2, 4, 5, 5,
  1, 4, 1, 3, 2, 6, 6, 4, 6, 4, 6, 4, 6, 3, 1, 1, 2, 6, 3, 2, 6, 6, 6, 3, 1]
```

Verification: checking that M is indeed expressed in this way.

```
> Id:=Matrix(2,2):
> Id[1,1]:=1: Id[2,2]:=1:
>
> t:=Id:
> for i to 574 do t:=Multiply(t,G[F[i]]): od:
> M+t;
                              [0  0]
                              [    ]
                              [0  0]
```

Second Preimage Attacks on Dithered Hash Functions

Elena Andreeva[1], Charles Bouillaguet[2], Pierre-Alain Fouque[2], Jonathan J. Hoch[3], John Kelsey[4], Adi Shamir[2,3], and Sebastien Zimmer[2]

[1] SCD-COSIC, Dept. of Electrical Engineering, Katholieke Universiteit Leuven Elena.Andreeva@esat.kuleuven.be

[2] École normale supérieure (Département d'Informatique), CNRS, INRIA
{Charles.Bouillaguet,Pierre-Alain.Fouque,Sebastien.Zimmer}@ens.fr

[3] Weizmann Institute of Science
{Adi.Shamir,Yaakov.Hoch}@weizmann.ac.il

[4] National Institute of Standards and Technology
john.kelsey@nist.gov

Abstract. We develop a new generic long-message second preimage attack, based on combining the techniques in the second preimage attacks of Dean [8] and Kelsey and Schneier [16] with the herding attack of Kelsey and Kohno [15]. We show that these generic attacks apply to hash functions using the Merkle-Damgård construction with only slightly more work than the previously known attack, but allow enormously more control of the contents of the second preimage found. Additionally, we show that our new attack applies to several hash function constructions which are not vulnerable to the previously known attack, including the dithered hash proposal of Rivest [25], Shoup's UOWHF[26] and the ROX hash construction [2]. We analyze the properties of the dithering sequence used in [25], and develop a time-memory tradeoff which allows us to apply our second preimage attack to a wide range of dithering sequences, including sequences which are much stronger than those in Rivest's proposals. Finally, we show that both the existing second preimage attacks [8,16] and our new attack can be applied even more efficiently to multiple target messages; in general, given a set of many target messages with a total of 2^R message blocks, these second preimage attacks can find a second preimage for one of those target messages with no more work than would be necessary to find a second preimage for a single target message of 2^R message blocks.

Keywords: Cryptanalysis, Hash Function, Dithering.

1 Introduction

A number of recent attacks on hash functions have highlighted weaknesses of both specific hash functions, and the general Merkle-Damgård construction. Wang *et al.* [28,29,30,31], Biham *et al.* [3], Klima [19] and Joux *et al.* [14] all show that differential attacks can be used to efficiently find collisions in specific

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 270–288, 2008.

hash functions based on the MD4 design, such as MD5, RIPEMD, SHA-0 and SHA-1. This type of result is important for at least two reasons. First, collision resistance is a required property for a hash function, and many applications of hash functions fail when collisions can be found. Second, efficiently found collisions permit additional attacks on hash functions using the Merkle-Damgård construction, as in Joux's [13] multicollision attack on cascade hashes, and the long-message second preimage attacks of Dean [8] and Kelsey and Schneier [16].

After Kelsey and Schneier published their attack, several researchers proposed a variant of the Merkle-Damgård construction, in which a third input to the compression function, called a "dithering sequence" in [25] and this paper, is used to block the attack. Specifically, using a dithering sequence prevents the construction of "expandable messages," required for both Dean and Kelsey and Schneier's attacks. In this paper, we develop a new kind of second preimage attack, which applies to some dithered variants of the Merkle-Damgård construction.

1.1 Related Work

The PhD thesis of Dean [8] presented a second preimage attack that works against a subset of hash functions using the Merkle-Damgård construction. Kelsey and Schneier [16] extended this result to work for *all* Merkle-Damgård hashes. For an n-bit hash function, their result allows an attacker to find a second preimage of a 2^k block[1] target message with $k \cdot 2^{n/2+1} + 2^{n-k}$ evaluations of the compression function. The attack relies on the ability to construct an *expandable message*, a set of incomplete messages of widely varying length, all of which yield the same intermediate hash result. This attack can be seen as a variant of the *long message attack* [20], in which the expandable message is used to carry out the attack despite the Merkle-Damgård strengthening.

Variants of the Merkle-Damgård construction that attempt to preclude the aforementioned second preimage attacks are the HAIFA [2] [23] construction proposed by Biham and Dunkelman and the "dithered" Merkle-Damgård hash by Rivest [25]. HAIFA includes the number of message bits hashed so far in the message block. The simplest way to implement HAIFA is to shorten each data block by 64 bits, filling those 64 bits with the 64 bit counter used internally to track the length of the hash input so far. Rivest, on the other hand, introduced a clever way to decrease the number of bits used for this extra input to either 2 or 16, thus increasing the bandwidth available for actual data, by using a specific sequence of values to "dither" the actual inputs. The properties of this sequence were claimed by Rivest to be sufficient to avoid the second preimage attack on the hash function.

The herding attack of Kelsey and Kohno [15] can be seen as another variant of the long-message attack. In their attack, the attacker first does a large precomputation, and then commits to a hash value h. Later, upon being challenged with

[1] In this paper, we describe message lengths in terms of message blocks, rather than bits. Most common hash functions use blocks of length 512 or 1024 bits.

[2] We do not have any attacks more efficient than exhaustive search on HAIFA.

a prefix P, the attacker constructs a suffix S such that hash$(P||S) = h$. Their paper introduced the "diamond structure", which is reminiscent of a complete binary tree. It is a 2^ℓ-multicollision in which each message in the multicollision has a different initial chaining value, and which is constructed in the precomputation step of the attack. The herding attack on an n-bit hash function requires approximately $2^{2n/3+1}$ work.

1.2 Our Results

In this paper, we develop a new generic second preimage attack on Merkle-Damgård hash functions and dithered Merkle-Damgård variants, treating the compression functions as black boxes. Our basic technique relies on the diamond from the herding attack of [15]. If the diamond is a 2^ℓ-multicollision, we obtain a second preimage of a message of length 2^k blocks with $2^{n/2+\ell/2+2} + 2^{n-\ell} + 2^{n-k}$ compression function computations. The attack is optimized when $\ell \approx n/3$, yielding an attack of complexity $5 \cdot 2^{2n/3} + 2^{n-k}$.

Our attack is slightly more expensive than the $k \cdot 2^{n/2+1} + 2^{n-k}$ complexity from [16] (for SHA-1, in which $n = 160$ and $k = 55$, the Kelsey-Schneier attack complexity is about 2^{105} work whereas ours is approximately 2^{109}). However, the new attack can be applied to Merkle-Damgård variants for which the attack of [16] is impossible. Our result also permits the attacker to leave most of the target message intact in the second preimage, or to arbitrarily choose the contents of roughly the first half of the second preimage, while leaving the remainder identical to the target message.

We can also apply our new second preimage attack to the dithered Merkle-Damgård hash variant of [25], exploiting the fact that the dithering sequences have many repetitions of some subsequences. For Rivest's proposed 16-bit dithering sequence, the attack requires $2^{n/2+\ell/2+2} + (8\ell + 32768) \cdot 2^{n-k} + 2^{n-\ell}$ work, which for SHA-1 is approximately 2^{120}. This is slightly worse than the attacks against the basic Merkle-Damgård construction but it is still much smaller than the 2^{160} security which was expected for the dithered construction. We show that the security of a dithered Merkle-Damgård hash is dependent on the number of distinct ℓ-letter subwords in the dithering sequence, and that the sequence chosen by Rivest is very susceptible to our attack.

We also show that the attack on dithered hashes is subject to a time-memory tradeoff that enables the construction of second preimages for any dithering input defined over a small alphabet with only a small amount of online computation after an expensive precomputation stage.

We further apply our attack to a one way hash function designed by Shoup [26], which has some similarities with dithered hashing. The attack applies as well to constructions that derive from this design, such as ROX [2]. Our technique yields the first published attack against these particular hash functions. This additionally proves that Shoup's security bound is tight, since there is asymptotically only a factor of $\mathcal{O}(k)$ between his bound and our attack's complexity.

Finally, we show that both the original second-preimage attack of [8,16] and our attack can be extended to the case in which there are multiple target messages. In general, finding a second preimage for any one of 2^t target messages of length 2^k blocks each requires approximately the same work as finding a single second preimage for a message of 2^{k+t} blocks.

1.3 Organization of the Paper

We describe our attack against the Merkle-Damgård construction in section 2. We introduce some terminology and describe the dithered Merkle-Damgård construction in section 3, and then we extend our attack to tackle dithered Merkle-Damgård in section 4. We apply it to Rivest's concrete proposal, as well as to some of the variations that he suggested. In section 5, we show that our attack works also against Shoup's UOWHF construction. We conclude with section 6, where we show how the second preimage attack may be applied to finding a second preimage for one of a large set of target messages.

2 A New Generic Second Preimage Attack

2.1 The Merkle-Damgård Construction

We first describe briefly the classical Merkle-Damgård construction. An iterated hash function $H^F : \{0,1\}^* \to \{0,1\}^n$ is built by iterating a basic compression function $F : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$. The hash process works as follows:

– Pad and split a message M into r blocks $x_1, \ldots, x_r$ of m bits each.
– Set h_0 to the initialization value IV.
– For each message block i compute $h_i = F(h_{i-1}, x_i)$.
– Output $H^F(M) = h_r$.

The padding is usually done by appending a single '1' bit followed by as many '0' bits as needed to complete an m-bit block. Merkle [21] and Damgård [7] independently proved in 1989 that making the binary encoding of the message length part of the padding improves the security of the construction: with this so-called *strengthening*, the scheme is proven to be Collision-Resistance Preserving, in the sense that a collision in the hash function H^F would imply a collision in the compression function F. As a side effect, the strengthening defines a limit over the maximal size of the messages that can be processed. In most deployed hash functions, this limit is 2^{64} bits, or equivalently 2^{55} 512-bit blocks. In the sequel, we denote the maximal number of admissible blocks by 2^k.

2.2 Second Preimage Attack on Merkle-Damgård Hash

We now describe a new technique to find second preimages on a Merkle-Damgård hash. It relies heavily on the "diamond structure" introduced by Kelsey and Kohno [15].

A diamond of size ℓ is a multicollision that has the shape of a complete converging binary tree of depth ℓ, with 2^ℓ leaves (hence we often refer to it as a *collision tree*). Its nodes are labelled by chaining values over n bits, and its edges are labelled by message blocks over m bits, which map between the chaining values at the two ends of the edge by the compression function. Thus, from any one of the 2^ℓ leaves, there is a path labelled by ℓ message blocks that leads to the same target value h_T labelling the root of the tree.

Let M be a target message of length 2^k blocks. The main idea of our attack is that connecting a message to a collision tree can be done in less than 2^n work. Moreover, connecting the root of the tree to one of the 2^k chaining values encountered during the computation of $H^F(M)$ takes only 2^{n-k} compression function calls. The attack works in 4 steps as described in figure 1.

1. Preprocessing step: compute a collision tree of depth ℓ with an arbitrary target value h_T. Note that this has to be done only once, and can be reused when computing second preimages of multiple messages.
2. Connect the target h_T to some chaining value in the message M. This can be done by generating random message blocks B, until $F(h_T, B) = h_{i_0}$ for some i_0, $\ell + 1 \leq i_0 < |M|$. Let B_0 be a message block satisfying this condition.
3. Generate an arbitrary prefix P of size $i_0 - \ell - 1$ blocks whose hash is one of the chaining values labelling a leaf. Let $h = H^F(P)$ be this value, and let T be the chain of ℓ blocks traversing the tree from h to h_T.
4. Form a message $M' = P||T||B_0||x_{i_0+1} \ldots x_{2^k}$.

Fig. 1. Summary of the attack on classic Merkle-Damgård

Messages M' and M are of equal length and hash to the same value, before strengthening, so they produce the same hash value despite the Merkle-Damgård strengthening.

A collision tree of depth ℓ can be constructed with time and space complexity $2^{\frac{n}{2}+\frac{\ell}{2}+2}$ (see [15] for details). The second step of the attack can be carried out with 2^{n-k} work, and the third one with $2^{n-\ell}$ work. The total time complexity of the attack is then: $2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2^{n-k} + 2^{n-\ell}$. This quantity becomes minimal when $\ell = (n-2)/3$, and in this setting, the total cost of our attack is about $5 \cdot 2^{2n/3} + 2^{n-k}$.

2.3 Comparison with Kelsey and Schneier

On the original Merkle-Damgård construction, the attack of [16] is more efficient than ours (on SHA-1, they can find a second preimage of a message of size 2^{55} with 2^{105} work, whereas we need 2^{109} calls to the compression function to obtain the same result).

However, our technique gives the adversary more control on the second preimage, since she can typically choose about the first half of the message in an arbitrary way. For example, she could choose to replicate most of the target message, leading to a second preimage that differs from the original by only $k+2$ blocks.

The main apparent difference between the two techniques is that the attack of Kelsey and Schneier relies on *expandable messages*. An expandable message $\mathcal{M}$ is a family of messages with different number of blocks but with the same hash when the final length block is not included in the computation. Their attack constructs such an expandable message in time $k \cdot 2^{n/2+1}$. Our attack can also be viewed as a new, more flexible technique to build expandable messages, by choosing a prefix of the appropriate length and connecting it to the collision tree. This can be done in time $2^{n/2+k/2+2} + 2^{n-k}$. Altough it is more expensive, this new technique can be adapted to work even when an additional dithering input is given, as we will demonstrate in the sequel.

3 Dithered Hashing

The general idea of dithered hashing is to perturb the hashing process by using an additional input to the compression function, formed by the consecutive elements of a fixed *dithering* sequence. This gives the attacker less control over the input of the compression function, and makes the hash of a message block dependent on its position in the whole message. In particular, the goal of dithering is to prevent attacks based on expandable messages.

Since the dithering sequence $\mathbf{z}$ has to be at least as long as the maximal number of blocks in any message that can be processed by the hash function, it is reasonable to consider infinite sequences as candidates for $\mathbf{z}$. Let $\mathcal{A}$ be a finite alphabet, and let the dithering sequence $\mathbf{z}$ be an eventually infinite word over $\mathcal{A}$. Let $\mathbf{z}[i]$ denote the i-th element of $\mathbf{z}$. The dithered Merkle-Damgård construction is obtained by setting $h_i = F\left(h_{i-1}, x_i, \mathbf{z}\left[i\right]\right)$ in the definition of the Merkle-Damgård scheme.

3.1 Words and Sequences

Notations and Terminology. Let ω be a word over the finite alphabet $\mathcal{A}$. The dot operator denotes concatenation. If ω can be written as $\omega = x.y.z$ (where x,y or z can be empty), we say that x is a *prefix* of ω and that y is a *factor* (or subword) of ω. A finite word ω is a *square* if it can be written as $\omega = x.x$, where x is not empty. A finite word ω is an *abelian square* if it can be written as $\omega = x.x'$ where x' is a permutation of x (*i.e.*, a reordering of the letters of x). A word is said to be *square-free* (resp. *abelian square-free*) if none of its factors is a square (resp. an abelian square). Note that abelian square-free words are also square-free.

An Infinite Abelian Square-Free Sequence. In 1992, Keränen [17] exhibited an infinite abelian square-free word $\mathbf{k}$ over a four-letter alphabet (there are no infinite abelian square-free words over a ternary alphabet). In this paper, we call this infinite abelian square-free word the *Keränen sequence*. Details about this construction can be found in [17,18,25].

Sequence Complexity. The number of factors of a given size of an infinite word gives an intuitive notion of its *complexity*: a sequence is more complex (or richer) if it possesses a large number of different factors. We denote by $Fact_{\mathbf{z}}(\ell)$ the number of factors of size ℓ of the sequence $\mathbf{z}$.

3.2 Rivest's Proposals

Keränen-DMD. Rivest suggested to directly use the Keränen sequence as a source of dithering inputs. The dithering inputs are taken from the alphabet $\mathcal{A} = \{a, b, c, d\}$, and can be encoded by two bits. The number of data bits in the input of the compression function is thus reduced by only two bits, which improves the hashing efficiency (compared to longer encodings of dither inputs). It is possible to generate the Keränen sequence online, one symbol at a time, in logarithmic space and constant amortized time.

Rivest's Concrete Proposal. Rivest's concrete proposal is referred to as DMD-CP (Dithered Merkle-Damgård – Concrete Proposal). To speed up the generation of the dithering sequence, Rivest proposed a slightly modified scheme, in which the dithering symbols are 16-bit wide. If the message M is r blocks long, then for $1 \leq i < r$ the i-th dithering symbol has the form:

$$\left(0, \mathbf{k}\left[\lfloor i/2^{13} \rfloor\right], i \mod 2^{13}\right) \in \{0,1\} \times \mathcal{A} \times \{0,1\}^{13}$$

The idea is to increment the counter for each dithering symbol, and to shift to the next letter in the Keränen sequence, only when the counter overflows. This "diluted" dithering sequence can essentially be generated 2^{13} times faster than the Keränen sequence. The last dithering symbol has a different form (recall that m is the number of bits in a message block):

$$(1, |M| \mod m) \in \{0,1\} \times \{0,1\}^{15}$$

4 Second Preimage Attacks on Dithered Merkle-Damgård

In this section, we present the first known second preimage attack on Rivest's dithered Merkle-Damgård construction. In section 4.1, we adapt the attack of section 2 to Keränen-DMD, obtaining second preimages in time $(k + 40.5) \cdot 2^{n-k+3}$. We then apply the extended attack to DMD-CP, obtaining second preimages with about 2^{n-k+15} evaluations of the compression function. We show some examples of sequences which make the corresponding dithered constructions immune to our attack. This notably covers the case of HAIFA [23]. Lastly, in section 4.2 we present a variation of the attack, which includes an expensive preprocessing, but which is able to cope with sequences of high complexity over a small alphabet with a very small online cost.

4.1 Adapting the Attack to Dithered Merkle-Damgård

Let us now assume that the hashing algorithm uses a dithering sequence $\mathbf{z}$. When building the collision tree, we must choose which dithering symbols to use.

A simple solution is to use the same dithering symbol for all the edges at the same depth in the tree. A tuple of ℓ letters is then required to build the collision tree. We will also need an additional letter to connect the tree to the message M. This way, in order to build a collision tree of depth ℓ, we have to fix a word ω of size $\ell+1$, use $\omega[i]$ as the dithering symbol of depth i, and use the last letter of ω to realize the connection.

The dithering sequence makes the hash of a block dependent on its position in the whole message. Therefore, the collision tree can be connected to its target only at certain positions, namely, at the positions where ω and $\mathbf{z}$ match. The set of positions in the message where this is possible is then given by:

$$\mathit{Range} = \left\{ i \in \mathbb{N} \,\middle|\, \big(\ell+1 \leq i\big) \wedge \big(\mathbf{z}[i-\ell] \dots \mathbf{z}[i] = \omega\big) \right\}.$$

Note that finding a connecting block B_0 in the second step defines the length of the prefix that is required. If $i_0 \in \mathit{Range}$, it will be possible to build the second preimage. Otherwise, another block B_0 has to be found.

To make sure that Range is not empty, ω has to be a factor of $\mathbf{z}$. Ideally, ω should be the factor of length $\ell+1$ which occurs most frequently in $\mathbf{z}$, as the cost of the attack ultimately depends on the number of connecting blocks tried before finding a useful one (with $i_0 \in \mathit{Range}$). What is the probability that a factor ω appears at a random position in $\mathbf{z}$? Although this is highly sequence-dependent, it is possible to give a generic lower bound: in the worst case, all factors of size $\ell+1$ appear in $\mathbf{z}$ with the same frequency. In this setting, the probability that a randomly chosen factor of size $\ell+1$ in $\mathbf{z}$ is the word ω is $1/\mathit{Fact}_{\mathbf{z}}(\ell+1)$.

The main property of $\mathbf{z}$ influencing the cost of our attack is its complexity (which is related to its min-entropy), whereas its repetition-freeness influences the cost of Kelsey and Schneier type attacks.

1. Choose the most frequent factor ω of $\mathbf{z}$, with $|\omega| = \ell+1$.
2. Build a collision tree of depth ℓ using ω as the dithering symbols in all the leaf-to-root paths. Let h_T be the target value of the tree.
3. Find a connecting block B_0 mapping h_T to anyone of the h_i (say h_{i_0}), by using $\omega[\ell]$ as the dithering letter. Repeat until $i_o \in \mathit{Range}$.
4. Carry the remaining steps of the attack as described in Fig. 1.

Fig. 2. Summary of the attack when a dithering sequence $\mathbf{z}$ is used

The cost of finding this second preimage for a given sequence $\mathbf{z}$, in the worst-case situation where all factors appear with the same frequency, is given by:

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + \mathit{Fact}_{\mathbf{z}}(\ell+1) \cdot 2^{n-k} + 2^{n-\ell}.$$

Cryptanalysis of Keränen-DMD. The cost of the extended attack against Keränen-DMD depends on the complexity of the sequence $\mathbf{k}$. Since it has a very regular structure, $\mathbf{k}$ has an unusually low complexity.

Lemma 1. *For $\ell \leq 85$, we have:*

$$Fact_{\mathbf{k}}(\ell) \leq 8 \cdot \ell + 332.$$

Despite being strongly repetition-free, the sequence **k** offers an extremely weak security level against our attack. We illustrate this by evaluating the cost of our attack on Keranen-DMD:

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + (8 \cdot \ell + 340) \cdot 2^{n-k} + 2^{n-\ell}.$$

If n is of the same order than about $3k$, then the first term of this sum is of the same order than the other two, and if $n \gg 3k$ then it can simply be neglected. We will use this approximation several times in the sequel. By setting $\ell = k - 3$, the total cost of the attack is about: $(k + 40.5) \cdot 2^{n-k+3}$ which is much smaller than 2^n in spite of the dithering.

Cryptanalysis of DMD-CP. We now apply the attack to Rivest's concrete proposal. We first need to evaluate the complexity of its dithering sequence. Recall from section 3.2 that it is based on the Keränen sequence, but that we move on to the next symbol of the sequence only when a 13 bit counter overflows. The original motivation was to reduce the cost of the dithering, but it has the unintentional effect of increasing the resulting sequence complexity. However, it is possible to prove that this effect is quite small:

Lemma 2. *Let* **c** *denote the sequence obtained by diluting* **k** *with a 13-bit counter. Then for every $0 \leq \ell < 2^{13}$, we have:*

$$Fact_{\mathbf{c}}(\ell) = 8 \cdot \ell + 32760.$$

The dilution does not generate a sequence of a higher asymptotic complexity: it is still linear in ℓ, even though the constant term is bigger due to the counter. The cost of the attack is therefore:

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + (8 \cdot \ell + 32768) \cdot 2^{n-k} + 2^{n-\ell}.$$

Again, if n is greater than about $3k$, the best value of ℓ is $k - 3$, and the complexity of the attack is then approximately: $(k + 4094) \cdot 2^{n-k+3} \simeq 2^{n-k+15}$. For settings corresponding to SHA-1, a second preimage can be computed in time 2^{120}.

Countermeasures. Even though the dilution does not increase the asymptotic complexity of a sequence, the presence of a counter increases the complexity of the attack. If we simply used a counter over i bits as the dithering sequence, the number of factors of size ℓ would be $Fact(\ell) = 2^i$ (as long as $i \leq \ell$). The complexity of the attack would then become: $2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2^{n-k+i} + 2^{n-\ell}$.

In practice, the dominating term is 2^{n-k+i}. By taking $i = k$, we would obtain a scheme which is resistant to our attack. This is essentially the choice made by the designers of Haifa [23], but such a dithering sequence consumes k bits of bandwidth. Note that as long as the counter does not overflow, no variation of the attack of Kelsey and Schneier can be applied to the dithered construction.

Using a counter (*i.e.*, a big alphabet) is a simple way to obtain a dithering sequence of high complexity. An other, somewhat orthogonal, possibility to improve the resistance of Rivest's dithered hashing to our attack is to use a dithering sequence of high complexity over a *small* alphabet (to preserve bandwidth). In appendix A we show that there is an abelian square-free sequence over 6 letters with complexity greater than $2^{\ell/2}$. Then, with $\ell = 2k/3$, the total cost of the online attack is about $2^{n-2k/3}$.

Another possible way to improve the resistance of Rivets's construction against our attack is to use a pseudo random sequence over a small alphabet. Even though it may not be repetition-free, its complexity is almost maximal. Suppose the alphabet has size $|\mathcal{A}| = 2^i$. Then the expected number of ℓ-letter factors in a pseudo random word of size 2^k is lower-bounded by: $2^{i\cdot\ell} \cdot \left(1 - \exp -2^{k-i\cdot\ell}\right)$ (refer to [12], theorem 2, for a proof of this claim)). The total optimal cost of the online attack is then at least $2^{n-k/(i+1)+2}$ and is obtained with $\ell = k/(i+1)$. With 8-bit dithering symbols and if $k = 55$, as in the SHA family, the complexity of the attack is 2^{n-5}.

4.2 A Generic Attack on Any Dithering Scheme with a Small Alphabet

The attacks described so far exploited the low complexity of Rivest's specific dithering sequences. In this section we show that the weakness is more general, and that after an $\mathcal{O}(2^n)$ preprocessing, second preimages can be found for messages of length $2^k \leq 2^{n/4}$ in $\mathcal{O}\left(2^{2\cdot(n-k)/3}\right)$ time and space for any dithering sequence (even of maximal complexity) if the dithering alphabet is small. Second preimages for longer messages can be found in $\max\left(\mathcal{O}\left(2^k\right), \mathcal{O}\left(2^{n/2}\right)\right)$ time and $\min\left(\mathcal{O}\left(2^{n-k}\right), \mathcal{O}\left(2^{n/2}\right)\right)$ memory.

Outline of the Attack. The new attack can be viewed as a type of time-memory tradeoff. For any given compression function, we precompute a fixed data structure which can then be used to find additional preimages for *any* dithering sequence and *any* given message of sufficient length. In the attack we will need to find connecting blocks leading from the message to our data structure and from our data structure to the message. The data structure will allow us to generate a sequence of blocks of the required length, leading from the entry point to the exit point, using the given dithering sequence.

A simple structure of this type is the *kite generator*[3] which will allow us to find a second preimage for a message made of $\mathcal{O}\left(2^k\right)$ message blocks in time $\max\left(\mathcal{O}\left(2^k\right), \mathcal{O}\left(2^{(n-k)/2}\right)\right)$ and $\mathcal{O}\left(|\mathcal{A}| \cdot 2^{n-k}\right)$ space. Note that for the SHA-1 parameters of $n = 160$ and $k = 55$, the time complexity of the new attack is 2^{55}, which is just the time needed to hash the original message. However, the size of the kite generator for the above parameters exceeds 2^{110}. The kite

[3] We call it a kite generator since we use it to generate *kites* of the form.

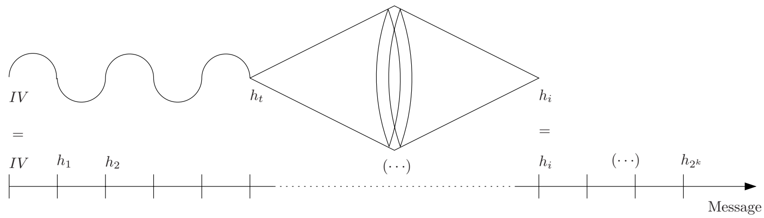


Fig. 3. A Kite

generator is a labelled directed graph whose 2^{n-k} vertices are labelled by some easily recognized subset of the chaining values that includes the IV (e.g., the tiny fraction of hash values which are extremely close to IV). Each directed edge (which can be traversed in both directions) is labelled by one letter α from the dithering alphabet and one message block x, and it leads from vertex h_1 to vertex h_2 if $F(h_1, x, \alpha) = h_2$. Each vertex in the generator should have exactly two outgoing edges labelled by each dithering letter, and thus the *expected* number of ingoing edges labelled by each letter is also 2. The generator is highly connected in the sense that there is an exponentially large diverging binary tree with any desired dithering sequence starting at any vertex, and an exponentially large converging tree [4] with any desired dithering sequence (whose degrees are not always 2) ending at most vertices. It can be viewed as a generalization of the collision tree of Kelsey and Kohno [15], which is a single tree with a single root in only the converging direction and with no dithering labels.

Once computed (during an unbounded precomputation stage), we can use the generator to find a second preimage for any given message M with 2^k blocks and any dithering sequence. We first hash the long input M to find (with high probability) some intermediate hash value h_i which appears in the generator. We then use the generator to replace the first i blocks in the message by a different set of i blocks. We start from the generator vertex labelled by IV, and follow some path in the generator of length $i - (n - k)$ which has the desired dithering sequence (there are exponentially many paths we can choose from). It leads to some hash value h_t in the generator. We then evaluate the full diverging tree of depth $(n - k)/2$ and the desired dithering sequence starting at h_t, and the full converging tree of depth $(n - k)/2$ and the desired dithering sequence ending at h_i. Since the number of leaves in each tree is $\mathcal{O}\left(2^{(n-k)/2}\right)$ and they are labelled by only 2^{n-k} possible values, we expect by the birthday paradox to find a common chaining value among the two sets of leaves. We can now combine the long random chain of length $i - (n - k)$ with the two short tree chains of length $(n - k)/2$ to find a *kite*-shaped structure of the same length i and with the same dithering sequence as the original message between the two chaining values IV and h_i. Note that the common leaf of the two trees can be found with no additional space by using a variant of Pollard's rho method which traverses pseudo-randomly chosen paths in the two trees until it cycles.

[4] See [10] for a formal justification of this claim.

This attack can be applied with essentially the same complexity even when the IV is not known during the precomputation stage (e.g., when it is time dependent). When we hash the original long message, we have to find two intermediate hash values h_i and h_j (instead of IV and h_i) which are contained in the generator and connect them by a properly dithered kite-shaped structure of the same length.

The main problem of this technique is that for the typical case in which $k < n/2$, it uses more space than time, and if we try to equalize them by reducing the size of the kite generator, we are unlikely to find any common chaining values between the given message and the generator. Finding a way to connect the generator back into the message will require 2^{n-k+1} additional steps, which will make the time complexity too high. To bypass this difficulty, we will use the classic time-memory tradeoff of Hellman tables.

Hellman's TMTO attack. Time/memory Tradeoffs (TMTO) were first introduced in 1980 by Hellman [11]. The idea is to improve brute force attacks by trading time for memory when inverting a function $f : \{0,1\}^n \to \{0,1\}^n$. Suppose we have an image element y and wish to find a pre-image $x \in f^{-1}(y)$. One extreme would be to go over all possible elements x until we find one such that $f(x) = y$, while the other extreme would be to pre-compute a huge table containing pairs $(x, f(x))$ sorted by the second element. Hellman's idea was to consider what happens when applying f iteratively. We start at a random element x_0 and compute $x_{i+1} = f(x_i)$ for t steps saving only the start and end points of the generated chain (x_0, x_t). We repeat this process with different initial points and generate a total of c chains. Now on input y we start generating a chain starting from y and check if we reach one of the saved endpoints. If we have, we generate the corresponding chain, starting from the original starting point and hope to find a preimage of y. Notice that as the number of chains c increases beyond $2^n/t^2$, the contribution from additional chains decreases with the number of chains. To counter this birthday paradox effect, Hellman suggested to construct a number of tables, each using a slightly different function f_i, such that knowing a preimage of y under f_i implies knowing such a preimage under f. Hellman's original suggestion, which works well in practice, was to use $f_i(x) = f(x \oplus i)$. Thus if we create $d = 2^{n/3}$ tables each with a different f_i, such that each table contains $c = 2^{n/3}$ chains of length $t = 2^{n/3}$, about 88% of the 2^n points will be covered by at least one table. Notice that the running time of Hellman's algorithm is $t \cdot d = 2^{2n/3}$ while the memory requirement is $d \cdot c = 2^{2n/3}$.

The Attack. As mentioned above, we need to find a linking block from the kite-generator to the message when its size is too small to have a common point. To solve this problem, we denote one of the vertices in the kite-generator by N and construct for each $\alpha \in \mathcal{A}$ a set of d Hellman tables with c chains, each of length t, such that $t \cdot c \cdot d = 2^{n-k}$ by iterating the basic function $f_\alpha(x) = F(N, x, \alpha)$. During the online phase, for each intermediate hash value h_i in the message, we use the set of tables corresponding to the dithering character α used to reach h_i

and try to find a block leading from the specified vertex N to h_i using α. Since the tables cover approximately 2^{n-k} elements, the probability of finding such a block for h_i is 2^{-k}. As the message is of length 2^k, we expect to find on average one connecting h_i. Notice that although we create chains for the Hellman tables, they do *not* correspond to the chain of hash values of a message, and thus we do not have to use the correct dithering sequence along these paths. The only purpose of the chains is to invert the function f_α and thereby find a single block linking N to one of the intermediate hash values along the given message.

Now that we have a method for connecting a predetermined hash value N to a message, we can replace the role of the kite-generator of finding a prefix which ends at N with a simpler construction. Since we were not constrained in our choice of N we can simplify the kite generator to the single point IV with a self loop for each dithering symbol $\alpha \in \mathcal{A}$. During the preprocessing, we exhaustively search for each $\alpha \in \mathcal{A}$ a block x_α such that $F(IV, x_\alpha, \alpha) = IV$. Given such self loops, we use in each step the block x_α corresponding to the current dithering symbol α and thus we can generate a message of any length starting and ending with IV. This IV serves as the point N in Hellman's algorithm. Note that this construction does not have the advantage of the original kite generator that IV can be unknown during the preprocessing stage.

Combining the two steps, we first find a linking block from IV to one of the intermediate hash values of the message using the correct dithering symbol. Then, using the IV self loops, we construct a prefix of the required length linking back to IV. During the preprocessing, the cost of constructing the Hellman tables is $|\mathcal{A}| \cdot t \cdot c \cdot d = \mathcal{O}\left(|\mathcal{A}| \cdot 2^{n-k}\right)$ time and $|\mathcal{A}| \cdot c \cdot d$ space, while constructing the IV self loops takes $\mathcal{O}\left(|\mathcal{A}| \cdot 2^n\right)$ time and $|\mathcal{A}|$ space. As the cost of finding the self loops is the dominating factor, the total time used in the preprocessing phase is $\mathcal{O}\left(|\mathcal{A}| \cdot 2^n\right)$ and the total space used is $|\mathcal{A}| \cdot c \cdot d$. In the online phase, generating the prefix takes time $\mathcal{O}\left(2^k\right)$ and finding a linking block to one of the 2^k intermediate hash values takes time $\mathcal{O}\left(2^k \cdot t \cdot d\right)$, so the total time spent in the online phase is $\mathcal{O}\left(2^k \cdot t \cdot d\right)$. For constant sized alphabets this leads to the following complexities: for $k \leq n/4$, a tradeoff balancing the time and memory costs is $t = 2^{(n-k)/3}, c = 2^{(n+2k)/3}, d = 2^{(n-4k)/3}$ giving total time and memory complexities of $\mathcal{O}\left(2^{2\cdot(n-k)/3}\right)$. For $n/4 < k \leq n/2$ the balanced time/memory tradeoff is achieved by using for each α a single table with parameters $c = 2^{n/2}$ and $t = 2^{n/2-k}$ giving a flat time and memory complexities of $\mathcal{O}\left(2^{n/2}\right)$. For a non-constant sized alphabet $\mathcal{A}$, the general time-memory tradeoff curve is $T \cdot M^2 \cdot 2^{2k} = 2^{2n} \cdot |\mathcal{A}|^2$ for $k \leq n/4$ and $T \geq 2^{2k}$.

5 An Attack on Shoup's UOWHF

In this section, we show that our attack is generic enough to be applied against hash functions enjoying a different security property, namely Universal One-Way Hash Functions (UOWHF). A UOWHF is a family of hash functions H for which any computationally bounded adversary A wins the following game with negligible probability. First A chooses a message M, then a key K is chosen at random and given to A. The adversary wins if she violates the Target Collision Resistance (TCR) of H, that is if she generates a message M' different from M that collides with M for the key K (*i.e.*, such that $H_K(M) = H_K(M')$ with $M \neq M'$).

Shoup [26] proposed a simple construction for a UOWHF that hashes messages of arbitrary size, given a UOWHF that hashes messages of fixed size. It is a Merkle-Damgård-like mode of operation, but before every iteration, one of several possible masks is XORed to the chaining value. The number of masks is logarithmic in the length of the hashed message, and the order in which they are used is carefully chosen to maximize the security of the scheme. This is reminiscent of dithered hashing, except that here the dithering process does not decrease the bandwidth available to actual data.

We first describe briefly Shoup's construction, and then show how our attack can be applied against it. The complexity of the attack demonstrates that for this particular construction, Shoup's security bound is nearly tight.

5.1 Description

This construction has some similarities with Rivest's dithered hashing. It starts from a universal one way compression function F that is keyed by a key K, $F_K\colon \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$. This compression function is then iterated, as described below, to obtain a variable input length UOWHF H_K^F.

The scheme uses a set of masks $\mu_0, \ldots, \mu_{k-1}$ (where $2^k - 1$ is the length of the longest possible message), each one of which is a random n-bit string. The key of the whole iterated function consists of K and of these masks. After each application of the compression function, a mask is XORed to the chaining value. The order in which the masks are applied is defined by a specified sequence over the alphabet $\mathcal{A} = \{0, \ldots, k-1\}$. The scheduling sequence is $\mathbf{z}[i] = \nu_2(i)$, for $1 \leq i \leq 2^k$, where $\nu_2(i)$ denotes the largest integer ν such that 2^ν divides i. Let M be a message that can be split into r blocks $x_1, \ldots, x_r$ and let h_0 be an arbitrary n-bit string. We define $h_i = F_K\left(h_{i-1} \oplus \mu_{\nu_2(i)}, x_i\right)$, and $H_K^F(M) = h_r$.

5.2 An Attack Matching the Security Bound

In [26], Shoup proves the following security result:

Theorem 1 (Main result of [26]). *If an adversary is able to break the target collision resistance of H^F with probability ϵ in time T, then one can construct an adversary that breaks the target collision resistance of F in time T, with probability $\epsilon/2^k$.*

In this section we show that this bound is almost tight. First, we give an alternate definition of the dithering sequence $\mathbf{z}$. We define:

$$u_i = \begin{cases} 0 & \text{if } i = 1, \\ u_{i-1}.(i-1).u_{i-1} & \text{otherwise.} \end{cases}$$

As an example, we have $u_4 = 010201030102010$. It is clear that $|u_i| = 2^i - 1$, and it is easy to show that for all i, u_i is a prefix of $\mathbf{z}$. The dithering sequence is thus simply u_k.

The most frequently-occurring factor of size $\ell < 2^k$ in $\mathbf{z}$ is the prefix of size ℓ of $\mathbf{z}$. It is a prefix of u_j with $j = \lceil \log_2 (\ell + 1) \rceil$, and u_j itself occurs about 2^{k-j} times in $\mathbf{z} = u_k$. The probability for a random factor of $\mathbf{z}$ of size ℓ to be exactly this candidate is equal to the number of occurrences of this candidate divided by the number of ℓ-bit strings in $\mathbf{z}$. Thus this probability is $\frac{2^{k-j}}{2^k - \ell}$. This can in turn be lower-bounded by: $2^{-j} \geq \frac{1}{2(\ell+1)}$. Our attack can be applied against the TCR property of H^F as described above. Choose at random a (long) target message M. Once the key is chosen at random, build a collision tree using a prefix of $\mathbf{z}$ of size ℓ for the dithering, and continue as described in section 4. The cost of the attack is then:

$$T = 2^{\frac{n}{2} + \frac{\ell}{2} + 2} + 2(\ell + 1) \cdot 2^{n-k} + 2^{n-\ell}.$$

This attack breaks the target collision resistance with probability nearly 1. Therefore, with Shoup's result, one can construct an adversary A against F with running time T and probability of success $1/2^k$. If F is a black box, the best attack against F's TCR property is the exhaustive search. Thus, the best attacker in time T against F has success probability $T/2^n$. When $n \geq 3k$, $T \simeq (2k+3) \cdot 2^{n-k}$ (with $\ell = k - 1$), and thus the best adversary running in time T has success probability $\mathcal{O}\left(k/2^k\right)$ when success probability of A is $1/2^k$. This implies that there is no attack better than ours by a factor greater than $\mathcal{O}(k)$ or, in other words, there is only a factor $\mathcal{O}(k)$ between Shoup's security proof and our attack.

The ROX construction by [2], which also uses the Shoup's mask sequence to XOR with the chaining values is susceptible to the same type of attack, which is also provably near-optimal.

5.3 Comparing the Shoup and Rivest Dithering Techniques

An intriguing connection between Shoup's and Rivest's ideas shows up as soon as we notice that the scheduling sequence $\mathbf{z}$ chosen by Shoup is abelian square-free. In fact, one year after Shoup's construction was published, Mironov [22] proved that an even stronger notion of repetition-freeness was necessary: $\mathbf{z}$ is, and has to be, *even-free*. A word is even-free if all of its non-empty factors contain at least one letter an odd number of times. Note that all even-free words are abelian square-free. We believe that the role these non-trivial sequences play in iterated constructions in cryptography (such as hashing) has yet to be completely understood.

6 Second Preimage Attack with Multiple Targets

Both the older generic second preimage results of [8,16] and our results can be applied efficiently to multiple target messages. The work needed for these attacks depends on the number of intermediate hash values of the target message, as this determines the work needed to find a linking message from the collision tree (our attack) or expandable message ([8,16]). A set of 2^R messages, each of 2^K blocks, has the same number of intermediate hash values as a single message of 2^{R+K} blocks, and so the difficulty of finding a second preimage for one of a set of 2^R such messages is no greater than that of finding a second preimage for a single 2^{R+K} block target message. In general, for the older second preimage attacks, the total work to find one second preimage falls linearly in the number of target messages; for our attack, it falls linearly so long as the total number of blocks 2^R satisfies $R < (n-4)/3$.

Consider for example an application which has used SHA-1 to hash 2^{30} different messages, each of 2^{20} message blocks. Finding a second preimage for a given one of these messages using the attack of [16] requires about 2^{141} work. However, finding a second preimage for *any one* of these of these 2^{30} target messages requires 2^{111} work. (Naturally, the attacker cannot control for *which* target message he finds a second preimage.)

This works because we can consider each intermediate hash value in each message as a potential target to which the root of the collision tree (or an expandable message) can be connected, regardless of the message it belongs to, and regardless of its length. Once we connect to an intermediate value, we have to determine to which particuliar target message it belongs. Then we can compute the second preimage of that message. Using similar logic, we can extend our attack on Rivest's dithered hashes, Shoup's UOWHF, and the ROX hash construction to apply to multiple target messages.

This observation is important for two reasons: First, simply restricting the length of messages processed by a hash function is not sufficient to block the long message attack; this is relevant for determining the necessary security parameters of future hash functions. Second, this observation allows long-message second preimage attacks to be applied to target messages of practical length. A second preimage attack which is feasible only for a message of 2^{50} blocks has no practical relevance, as there are probably no applications which use messages of that length. A second preimage attack which can be applied to a large set of messages of, say, 2^{24} blocks, might have some practical impact. While the computational requirements of these attacks are still infeasible, this observation shows that the attacks can apply to messages of practical length.

Acknowledgments. Thanks to Orr Dunkelman for many useful suggestions and discussions, to Lily Chen and Barbara Guttman for useful comments, and to J.-P. Allouche, J. Shallit and J. Curie for pointing out the existence of abelian square-free sequences of high complexity.

The work of the first author has been funded by a Ph.D. grant of the Flemish Research Foundation and supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy). This work is partly supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

1. Allouche, J.-P.: Sur la complexité des suites infinies. Bull. Belg. Math. Soc. 1, 133–143 (1994)
2. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-Property-Preserving Iterated Hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
3. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and Reduced SHA-1. In: [16], pp. 36–57
4. Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
5. Cobham, A.: Uniform tag seqences. Mathematical Systems Theory 6(3), 164–192 (1972)
6. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
7. Damgård, I.: A Design Principle for Hash Functions. In: [4], pp. 416–427
8. Dean, R.D.: Formal Aspects of Mobile Code Security. PhD thesis, Princeton University (January 1999)
9. Ehrenfeucht, A., Lee, K.P., Rozenberg, G.: Subword Complexities of Various Classes of Deterministic Developmental Languages without Interactions. Theor. Comput. Sci. 1(1), 59–75 (1975)
10. Feller, W.: 12. In: An Introduction to Probability Theory and Its Applications, vol. 1, John Wiley & sons, Chichester (1971)
11. Hellman, M.E.: A cryptanalytic time-memory trade off. In: IEEE Transactions on Information Theory, vol. IT-26, pp. 401–406 (1980)
12. Janson, S., Lonardi, S., Szpankowski, W.: On average sequence complexity. Theor. Comput. Sci. 326(1-3), 213–227 (2004)
13. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
14. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
15. Kelsey, J., Kohno, T.: Herding Hash Functions and the Nostradamus Attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
16. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: [6], pp. 474–490
17. Keränen, V.: Abelian Squares are Avoidable on 4 Letters. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 41–52. Springer, Heidelberg (1992)
18. Keränen, V.: On abelian square-free DT0L-languages over 4 letters.. In: Harju, T. (ed.) WORDS 2003, vol. 27, pp. 95–109. TUCS General Publication (2003)
19. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105 (2006) http://eprint.iacr.org/
20. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography
21. Merkle, R.C.: One Way Hash Functions and DES. In: [4], pp. 428–446
22. Mironov, I.: Hash Functions: From Merkle-Damgård to Shoup. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 166–181. Springer, Heidelberg (2001)
23. Orr Dunkelman, E.B.: A Framework for Iterative Hash Functions — HAIFA. Presented at the second NIST hash workshop (August 24–25, 2006)

24. Pansiot, J.-J.: Complexité des facteurs des mots infinis engendrés par morphismes itérés. In: Paredaens, J. (ed.) ICALP 1984. LNCS, vol. 172, pp. 380–389. Springer, Heidelberg (1984)
25. Rivest, R.L.: Abelian Square-Free Dithering for Iterated Hash Functions. In: Presented at ECrypt Hash Function Workshop, June 21, 2005, Cracow, and at the Cryptographic Hash workshop, November 1, 2005, Gaithersburg, Maryland (August 2005)
26. Shoup, V.: A Composition Theorem for Universal One-Way Hash Functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
27. Shoup, V. (ed.): CRYPTO 2005. LNCS, vol. 3621. Springer, Heidelberg (2005)
28. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: [6], pp. 1–18
29. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: [27], pp. 17–36
30. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: [6], pp. 19–35
31. Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: [27], pp. 1–16

A Some Sequence-Complexity Related Results

Sequences Generated by Morphisms. We say that a function $\tau : \mathcal{A}^* \to \mathcal{A}^*$ is a *morphism* if for all words x and y, $\tau(x.y) = \tau(x).\tau(y)$. A morphism is then entirely determined by the images of the individuals letters. A morphism is said to be r-*uniform* (with $r \in \mathbb{N}$) if for all word x, $|\tau(x)| = r \cdot |x|$. If, for a given letter $\alpha \in \mathcal{A}$, we have $\tau(\alpha) = \alpha.x$ for some word x, then τ is *non-erasing* for α. Given a morphism τ and an initialization letter α, let u_n denote the n-th iterate of τ over α: $u_n = \tau^n(\alpha)$. If τ is r-uniform (with $r \geq 2$) and non-erasing for α, then u_n is a strict prefix of u_{n+1}, for all $n \in \mathbb{N}$. Let $\tau^\infty(\alpha)$ denote the limit of this sequence: it is the only fixed point of τ that begins with the letter α. Such infinite sequences are called *uniform tag sequences* [5] or r-*automatic* sequences [1]. Because they have a very regular structure, there is a spectacular result [5] regarding the complexity of infinite sequences generated by uniform morphisms:

Theorem 2 (Cobham, 1972). *Let* $\mathbf{z}$ *be an infinite sequence generated by an* r*-uniform morphism, and assume that the alphabet size* $|\mathcal{A}|$ *is constant. Then* $\mathbf{z}$ *has* linear *complexity:*

$$Fact_{\mathbf{z}}(\ell) \leq r \cdot |\mathcal{A}|^2 \cdot \ell.$$

It is worth mentioning that similar results exist in the case of sequences generated by non-uniform morphisms [24,9], although the upper bound can be quadratic in ℓ. Since the Kerañen sequence is 85-uniform [17,18,25], the result of theorem 2 gives: $Fact_{\mathbf{k}}(\ell) \leq 1360 \cdot \ell$. This upper-bound is relatively rough, and for particular values of ℓ, it is possible to obtain a much better approximation, such as the one given in lemma 1 (which is tight). The interested reader should consult the full version of this paper.

There are Abelian Square-Free Sequences of Exponential Complexity. It is indeed possible to construct an infinite abelian square-free sequence of exponential complexity, although we do not know how to do it without slightly enlarging the alphabet.

We start with the abelian square-free Keraïen sequence $\mathbf{k}$ over $\{a, b, c, d\}$, and with another sequence $\mathbf{u}$ over $\{0, 1\}$ that has an exponential complexity. Such a sequence can be built for example by concatenating the binary encoding of all the consecutive integers. Then we can create a sequence $\tilde{\mathbf{z}}$ over the union alphabet $\mathcal{A} = \{a, b, c, d, 0, 1\}$ by interleaving $\mathbf{k}$ and $\mathbf{u}$: $\tilde{\mathbf{z}} = \mathbf{k}[1].\mathbf{u}[1].\mathbf{k}[2].\mathbf{u}[2].\dots$. The resulting shuffled sequence inherits both properties: it is still abelian square-free, and has a complexity of order $\Omega\left(2^{\ell/2}\right)$.

Efficient Two Party and Multi Party Computation Against Covert Adversaries

Vipul Goyal[1], Payman Mohassel[2], and Adam Smith[3]

[1] Department of Computer Science, UCLA
vipul@cs.ucla.edu
[2] Department of Computer Science, UC Davis
pmohassel@ucdavis.edu
[3] Department of Computer Science, PSU
asmith@cse.psu.edu

Abstract. Recently, Aumann and Lindell introduced a new realistic security model for secure computation, namely, security against *covert adversaries*. The main motivation was to obtain secure computation protocols which are efficient enough to be usable in practice. Aumann and Lindell presented an efficient two party computation protocol secure against covert adversaries. They were able to utilize cut and choose techniques rather than relying on expensive zero knowledge proofs.

In this paper, we design an efficient multi-party computation protocol in the covert adversary model which remains secure even if a majority of the parties are dishonest. We also substantially improve the two-party protocol of Aumann and Lindell. Our protocols avoid general NP-reductions and only make a *black box* use of efficiently implementable cryptographic primitives. Our two-party protocol is constant-round while the multi-party one requires a logarithmic (in number of parties) number of rounds of interaction between the parties. Our protocols are secure as per the standard *simulation-based* definitions of security.

Although our main focus is on designing efficient protocols in the covert adversary model, the techniques used in our two party case directly generalize to improve the efficiency of two party computation protocols secure against standard malicious adversaries.

1 Introduction

Secure multi-party computation (MPC) allows a set of n parties to compute a joint function of their inputs while keeping their inputs private. General secure MPC has been an early success of modern cryptography through works such as [Yao86, GMW87, BOGW88, CCD88]. The early MPC protocols used very generic techniques and were inefficient. Hence, now that most of the questions regarding the *feasibility* of secure computation have been addressed (at least in the stand alone setting), many of the recent works have focused on improving the efficiency of these protocols.

The most hostile situation where one could hope to do secure computation is when we have a *dishonest majority*. That is, where up to $(n-1)$ parties could

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 289–306, 2008.

be corrupted and could deviate arbitrarily from the protocol. The feasibility of secure computation in this setting was shown by [GMW87]. Several later results focused on improving its efficiency (often quantified as round complexity).

Most of these constructions use *general zero-knowledge proofs* to compile honest-but-curious MPC protocols into fully malicious MPC protocols. These zero-knowledge compilers are of great theoretical importance but lead to rather inefficient constructions. These compilers make a *non-black-box* use of the underlying cryptographic primitives. To illustrate this inefficiency, consider the following example taken from [IKLP06]. Suppose that due to major advances in cryptanalytic techniques, all basic cryptographic primitives require a full second of computation on a fast CPU. *Non-black-box* constructions require parties to prove in zero-knowledge, statements that involve the computation of the underlying primitives, say a trapdoor permutation. These zero-knowledge protocols, in turn, invoke cryptographic primitives for every gate of a circuit computing a trapdoor permutation. Since (by our assumption) a trapdoor permutation takes one second to compute, its circuit implementation contains trillions of gates, thereby requiring the protocol trillions of second to run. A black box construction, on the other hand, would make the number of invocations of the primitive independent of the complexity of implementing the primitive.

Due to lack of efficient and practical constructions for the case of dishonest majority, a natural question that arises is *"Can we relax the model (while still keeping it meaningful) in a way which allows us to obtain efficient protocols likely to be useful in practice?"*.

One such model is the well known honest majority model. The model additionally allows for the construction of protocols with guaranteed output delivery. Positive steps to achieve efficient protocols in this model were taken by Damgard and Ishai [DI05]. They presented an efficient protocol which makes a black box use of only a pseudorandom generator.

Another such model is the model of *covert adversaries* (incomparable to the model of honest majority) recently introduced by Aumann and Lindell[AL07] (see also [CO99]). A covert adversary may deviate from steps of the protocol in an attempt to cheat, but such deviations are detected by honest parties with good probability (although not with negligibly close to 1). As Aumann and Lindell argue, covert adversaries model many real-world settings where adversaries are willing to actively cheat (and therefore are not semi-honest) but only if they are not caught doing so. This is the case for many business, financial, political and diplomatic settings where honest behavior cannot be assumed but where companies, institutions, or individuals cannot afford the embarrassment, loss of reputation and negative press associated with being caught cheating. They further proceed to design an efficient two-party computation protocol secure against covert adversaries with only blackbox access to the underlying primitives. Their construction applies cut-and-choose techniques to Yao's garbled circuit, and takes advantage of an efficient oblivious transfer protocol secure against covert adversaries. Currently, there is no such counterpart for the case of ≥ 3 parties with dishonest majority.

Our Results

Multi-party Computation against Covert Adversaries. We construct a protocol for multi-party computation in the covert adversary model. Our protocol provides standard simulation based security guarantee if any number of the parties collude maliciously. Our techniques rely on efficient cut and choose techniques and avoid expensive zero-knowledge proofs to move from honest-but-curious to malicious security. We only make a *black-box use* of *efficiently implementable* cryptographic primitives.

The protocol requires $O(n^3 ts|C|)$ bits of communication (and similar computation time) to securely evaluate a circuit C with deterrence $1 - \frac{1}{t}$. Here $\frac{1}{t}$ is the noticeable, but small probability with which the cheating parties may escape detection, and s is a cryptographic security parameter. In contrast, the most efficient previously known protocols, due to Katz, Ostrovsky and Smith [KOS03] and Pass [Pas04], require zero-knowledge proofs *about* circuits of size $O(n^3 s|C|)$.

The protocol in this paper requires $O(\log n)$ rounds of interaction, due to an initial coin-flipping phase that follows the Chor-Rabin scheduling paradigm [CR87]. The round complexity can be reduced to a constant using non-black-box simulation techniques [Bar02,KOS03,Pas04], but the corresponding increase in computational complexity makes it unlikely that the resulting protocol would be practical.

We remark that there have been a number of two-parties protocols designed using cut and choose techniques [MNPS04,MF06,Woo07,LP07], where one party prepares several garbled circuits while the other party randomly checks a subset of them. However, this paper is the first work to employ such techniques for the design of efficient protocols in the multi-party setting.

Two-party Computation against Covert Adversaries. In a protocol secure against covert adversaries, any attempts to cheat by an adversary is detected by honest parties with probability at least ϵ, where ϵ is the deterrence probability. Therefore, a *high deterrence probability* is crucial in making the model of covert adversaries a practical/realistic model for real-world applications. In this paper we design a *two-party protocol secure against covert adversaries* in which the deterrence probability $\epsilon = 1 - 1/t$, for any value of t polynomial in the security parameter, comes almost for *free* in terms of the *communication complexity* of the protocol. The following table compares our result against that of previous work, where $|C|$ is the circuit size, m is the input size, and s is the statistical security parameter.

Protocol	Communication Complexity
[AL07]	$O(t\|C\| + tsm)$
This paper (section 3.1)	$O(\|C\| + sm + t)$

Two-party Computation against Fully Malicious Adversaries. Although we mainly focus on covert adversaries, we also show how our techniques lead to secure two-party computation schemes against *fully malicious* adversaries. Particularly, by applying our techniques to the existing cut-and-choose protocols, i.e. [LP07,Woo07,MF06], we improve the communication cost of these protocols without affecting their security guarantees. In this case, our improvement in the communication cost of these protocols is not asymptotic but rather in concrete terms.

Related Work. Katz *et al.* [KOS03] and Pass [Pas04] give the most round-efficient secure MPC protocols with dishonest majority. Ishai et al. [IKLP06], give the first construction for dishonest majority with only black-box access to a trapdoor permutation. Although theoretically very interesting, these approaches are not attractive in terms of efficiency due to the usage of very generic complexity theoretic techniques.

The compiler of Lindell [Lin01] may be applied to achieve constant-round protocols for secure two-party computation. More recent works on secure two-party computation avoid the zero-knowledge machinery (using cut-and-choose techniques), and design efficient protocols with only black-box access to the underlying primitives. Application of cut-and-choose techniques to Yao's garbled circuit was first suggested by Pinkas [Pin03], and further refined and extended in [MNPS04,MF06,Woo07,LP07]. The protocols of [MF06] and [LP07] lead to $O(s|C|+s^2m)$ communication between the parties, while the protocol of [Woo07] only requires $O(s|C|)$ communication where s is the security parameter. Our improvement in the communication cost of these protocols is not asymptotic but rather in concrete terms. Lindell and Pinkas [LP07] also showed how the cut-and-choose techniques could be modified to also yield simulation-based proofs of security. Their ideas can also be applied to [MF06,Woo07]. A different approach for defending against malicious adversaries in two party computation is taken by Jarecki and Shmatikov [JS07]. The basic idea in their work is to have the first party generate a garbled circuit and prove its correctness by giving an efficient number-theoretic zero-knowledge proof of correctness for every gate in the circuit. This protocol is more communication efficient than the cut-and-choose schemes, but increases the computational burden of the parties. In particular, the protocol of [JS07] requires $O(|C|)$ public-key operations while the cut-and-choose schemes only require $O(m)$ public-key operations. As shown in experiments (e.g. see [MNPS04]) the public-key operations tend to be the computational bottle-neck in practice.

The idea of allowing the adversary to cheat as long as it will be detected with a reasonable probability was first considered in [FY92] under the term t-detectability. Work of [FY92] only considers honest majority and the definition is not simulation based. Canetti and Ostrovsky [CO99] consider *honest-looking adversaries* who may deviate arbitrarily form the protocol specification as long as the deviation cannot be detected. [AL07] introduce the notion of covert adversaries which is similar in nature to the previous works but strengthens them in several ways. The most notable are that it quantifies over all possible adversaries (as opposed to adversaries that behave in a certain way), and puts the burden of detection of cheating on the protocol, and not on the honest parties analyzing the transcript distribution later on.

2 Preliminaries

2.1 Definition of Security Against Covert Adversaries

Aumann and Lindell, [AL07], give a formal definition of security against covert adversaries in the *ideal/real simulation paradigm*. This notion of adversary lies somewhere between those of semi-honest and malicious adversaries. Loosely speaking, the definition provides the following guarantee: Let $0 \leq \epsilon \leq 1$ be a value (called the deterrence factor). Then any attempts to cheat by an adversary is detected by the honest parties with probability at least ϵ. Thus provided that ϵ is sufficiently large, an adversary that wishes not to get caught cheating will refrain from attempting to cheat, lest it be caught doing so. Furthermore, in the strongest version of security against covert adversaries introduced in [AL07], the adversary will not learn any information about the honest parties' inputs if he gets caught. What follows next is the strongest version of their definition (which is what we use as the security definition for all of our protocols) and is directly taken from [AL07]. The executions in the real and ideal model are as follows:

Execution in the real model. Let the set of parties be $P_1, \ldots, P_n$ and let $\mathcal{I} \subset [n]$ denote the indices of corrupted parties, controlled by an adversary $\mathcal{A}$. We consider the real model in which a real n-party protocol π is executed (and there exist no trusted third party). In this case, the adversary $\mathcal{A}$ sends all messages in place of corrupted parties, and may follow an arbitrary polynomial-time strategy. In contrast, the honest parties follow the instructions of π.

Let $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$ be an n-party functionality where $f = (f_1, \ldots, f_n)$, and let π be an n-party protocol for computing f. Furthermore, let $\mathcal{A}$ be a non-uniform probabilist polynomial-time machine and let $\mathcal{I}$ be the set of corrupted parties. Then the real execution of π on inputs $\bar{x}$, auxiliary input z to $\mathcal{A}$ and security parameter s, denoted $REAL_{\pi,\mathcal{A}(z),\mathcal{I}}(\bar{x}, s)$, is defined as the output vector of the honest parties and the adversary $\mathcal{A}$ from the real execution of π.

Execution in the Ideal Model. Let $\epsilon : \mathcal{N} \rightarrow [0, 1]$ be a function. Then the ideal execution with ϵ proceeds as follows.

Inputs: Each party obtains an input; the i^{th} party's input is denoted by x_i; we assume that all inputs are of the same length m. The adversary receives an auxiliary-input z.

Send inputs to trusted party: Any honest party P_j sends its received input x_j to the trusted party. The corrupted parties, controlled by $\mathcal{A}$, may either send their received input or send some other input of the same length to the trusted party. This decision is made by $\mathcal{A}$ and may depend on x_i for $i \in \mathcal{I}$ and the auxiliary input z. Denote the vector of inputs sent to the trusted party by $\bar{w}$.

Abort Options: If a corrupted party sends $w_i = \mathsf{abort}_i$ to the trusted party as its input, then the trusted party sends abort_i to all of the honest parties and halts. If a corrupted party sends $w_i = \mathsf{corrupted}_i$ as its input to the trusted party, then the trusted party sends $\mathsf{corrupted}_i$ to all of the honest parties and halts.

Attempted cheat option: If a corrupted party sends $w_i = \mathsf{cheat}_i$ to the trusted party as its input, then:

1. With probability $1 - \epsilon$, the trusted party sends $\mathsf{corrupted}_i$ to the adversary and all of the honest parties.
2. With probability ϵ, the trusted party sends $\mathsf{undetected}$ and all of the honest parties inputs $\{x_j\}_{j \notin \mathcal{I}}$ to the adversary. The trusted party asks the adversary for outputs $\{y_j\}_{j \notin \mathcal{I}}$, and sends them to the honest parties.

The ideal execution then ends at this point. If no w_i equals abort_i, $\mathsf{corrupted}_i$ or cheat_i the ideal execution continues below.

Trusted party answers adversary: The trusted party computes $(f_1(\bar{w}), \ldots, f_m(\bar{w}))$ and sends $f_i(\bar{w})$ to $\mathcal{A}$, for all $i \in \mathcal{I}$.

Trusted party answers honest parties: After receiving its outputs, the adversary sends either abort_i for some $i \in \mathcal{I}$ or $\mathsf{continue}$ to the trusted party. If the trusted party receives the $\mathsf{continue}$ then it sends $f_i(\bar{w})$ to all honest parties $P_j (j \notin \mathcal{I})$. Otherwise, if it receives abort_i for some $i \in \mathcal{I}$, it sends abort_i to all honest parties.

Outputs: An honest party always outputs the messages it obtained from the trusted party. The corrupted parties output nothing. The adversary $\mathcal{A}$ outputs any arbitrary (probabilistic polynomial-time computable) function of the initial inputs $\{x_i\}_{i \in \mathcal{I}}$ and messages obtained from the trusted party.

The output of honest parties and the adversary in an execution of the above model is denoted by $IDEAL^{\epsilon}_{f,S(z),\mathcal{I}}(\bar{x}, s)$ where s is the statistical security parameter.

Definition 1. *Let f, π, ϵ be as described above. Protocol π is said to* securely compute f in the presence of covert adversaries with $\epsilon-$deterrence *if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ for the real model, there exist a non-uniform probabilistic polynomial-time adversary S for the ideal model such that for every $\mathcal{I} \subseteq [n]$, every balanced vector $\bar{x} \in (\{0,1\}^*)^n$, and every auxiliary input $z \in \{0,1\}^*$:*

$$IDEAL^{\epsilon}_{f,S(z),\mathcal{I}}(\bar{x}, s) \stackrel{c}{\equiv} REAL_{\pi,\mathcal{A}(z),\mathcal{I}}(\bar{x}, s)$$

3 The Two Party Case

3.1 Efficient Two Party Computation for Covert Adversaries

Aumann and Lindell [AL07] design an efficient two-party computation protocol secure against covert adversaries. In their protocol, two parties P_1 and P_2 wish to securely compute a circuit C that computes a function f on parties private inputs. The high level idea of their protocol is that party P_1 computes t garbled circuits[1], and sends them to party P_2. P_2 then randomly chooses one circuit to compute and asks P_1 to reveal the secrets of the remaining $(t-1)$ circuits. This ensures that a cheating P_1 gets caught with probability at least equal to $1-1/t$. There are other subtleties in order to deal with parties' inputs and to achieve simulation-based security. We will go into more detail regarding these subtleties later in this section. Aumann and Lindell also design a special and highly efficient oblivious transfer protocol secure against covert adversaries which makes their solution even more practical. The efficiency of their protocol can be summarized in the following statement ($|C|$ is the circuit size, m is the input size and s is the security parameter):

Theorem 1. *([AL07]) There exist a two-party computation protocol secure against covert adversaries with deterrence value* $1-1/t$ *such that the protocol runs in a constant number of rounds, and requires* $O(t|C|+tsm)$ *communication between the two players.*

Our Protocol. We now design a secure two-party computation protocol in presence of covert adversaries for which the deterrence probability $1-1/t$, for any value of t polynomial in the security parameter, comes almost for free in terms of the *communication complexity of the protocol* (assuming the circuit being evaluated is large enough). In the remiander of the paper, we assume familiarity with the Yao's garbled circuit protocol.

We first observe that for the simulation-based proof of the protocol to go through and for the simulator to be able to extract corrupted P_2's inputs, it is not necessary to run the complete oblivious transfers early in the protocol for all the garbled circuits. Instead, it is enough to go as far in the steps of the OTs as is necessary for party P_2 to be committed to his input bits while party P_1 is still free to choose his inputs to the OT. Parties then postpone the remaining steps of the OTs until later in the protocol when one circuit among the t garbled circuits is chosen to be evaluated. With some care, this leads to asymptotic improvement in communication complexity of our protocol.

To achieve further improvement in communication complexity, we take a different approach to constructing the garbled circuit. In order to compute a garbled circuit (and the commitments for input keys), party P_1 generates a short random seed and feeds it to a pseudorandom generator in order to generate the necessary randomness. He then uses the randomness to construct the garbled circuit and the necessary commitments. When the protocol starts, party P_1 sends to P_2 only a hash of each garbled circuit using a collision-resistant hash function. Later in the protocol, in order to expose the secrets of each circuit, party P_1 can simply send the seeds corresponding to that circuit to P_2, and not the whole opened circuit. In the full version of this paper, we describe in more detail, how to generate the garbled circuit in this way.

Before describing the details of our protocol, it is helpful to review a trick introduced by [LP07] for preventing a subtle malicious behavior by a corrupted

[1] The garbled circuits are constructed according to Yao's garbled circuit protocol(see [LP04] for a detailed explanation).

P_1. For instance, during an oblivious transfer protocol, a corrupted P_1 can use an invalid string for the key associated with value 0 for P_2's input bit but a valid string for the key associated with 1. An honest P_2 is bound to abort if any of the keys he receives are invalid. But the action P_2 takes reveals his input bit to P_1. To avoid this problem, we use a circuit that computes the function $g(x_1, x_2^1, \ldots, x_2^s) = f(x_1, \oplus_{i=1}^s x_2^i)$ instead of a circuit that directly computes f. For his actual input x_2, party P_2 chooses s random inputs $x_2^1, \ldots, x_2^s$ such that $x_2 = x_2^1 \oplus \ldots \oplus x_2^s$. This solves the problem since for P_1 to learn any information about P_2's input he has to send invalid keys for all s shares. But, if P_1 attempts to give invalid key for all s shares of P_2's input, he will get caught with exponentially high probability in s. We are now ready to describe our protocol. We borrow some of our notations from [LP04] and [AL07].

The Protocol

Party P_1's input: x_1
Party P_2's input: x_2
Common input: Both parties have security parameter m; for simplicity let $|x_1| = |x_2| = m$. Parties agree on the description of a circuit C for inputs of length m that computes function f. P_2 chooses a collision-resistant hash function h. Parties agree on a pseudorandom generator G, a garbling algorithm $Garble$, a perfectly binding commitment scheme Com_b, and a deterrence probability $1 - 1/t$.

1. Parties P_1 and P_2 define a new circuit C' that receives $s + 1$ inputs x_1, $x_2^1, \ldots, x_2^s$ each of length m, and computes the function $f(x_1, \oplus_{i=1}^s x_2^i)$. Note that C' has $m(s+1)$ input wires. Denote the input wires associated with x_1 by $w_1, \ldots, w_m$ and the input wires associated with x_2^i by $w_{im+1}, \ldots, w_{im+m}$ for $i = 1, \ldots, s$.
2. Party P_2 chooses $(s-1)$ random strings $x_2^1, \ldots, x_2^{s-1} \in_R \{0,1\}^m$ and defines $x_2^s = (\oplus_{i=1}^{s-1} x_2^i) \oplus x_2$. The value $z_2 = (x_2^1, \ldots, x_2^s)$ serves as P_2's new input of length sm to C'.
3. Parties perform the first four steps of the OT protocol of [AL07] for P_2's sm input bits (see the full version for more detail). [2]
4. Party P_1 generates t random seeds $s_1, \ldots, s_t$ of appropriate length and computes $GC_i = Garble(G, s_i, C')$ for $1 \leq i \leq t$ (see the full version of this paper for $Garble()$ algorithm). He then sends $h(GC_1)), \ldots, h(GC_t)$ to P_2.
5. P_1 generates t random seeds $s'_1, \ldots, s'_t$ of appropriate length and computes $G(s'_i)$ from which he extracts the randomness $r_j^{b,i}$ (later used to construct a commitment) for every $1 \leq i \leq t$, every $j \in \{1, \ldots, sm + m\}$, and every $b \in \{0, 1\}$, and the random order for the commitments to keys for his own input wires (see next step). He then computes the commitments

[2] Any other constant-round oblivious transfer protocol secure against covert adversaries with the property that– there exists an step in the protocol where P_2 is committed to his input while P_1 is still free to choose his input– can be used here as well.

$c_j^{b,i} = Com_b(k_j^{b,i}, r_j^{b,i})$ for every $i \in \{1,\ldots,t\}$, every $j \in \{1,\ldots,sm+m\}$, and every $b \in \{0,1\}$.

6. For every $1 \leq i \leq t$, P_1 computes two sets A_i and B_i, consisting of pairs of commitments. The order of each pair in B_i is chosen at random (using the randomness generated by $G(s'_i)$), but the order of each pair in A_i is deterministic, i.e., commitment to the key corresponding to 0 comes before the one corresponding to 1.

$$A_i = \{(c_{m+1}^{0,i}, c_{m+1}^{1,i}), \ldots, (c_{m+sm}^{0,i}, c_{m+sm}^{1,i})\}$$
$$B_i = \{(c_1^{0,i}, c_1^{1,i}), \ldots, (c_m^{1,i}, c_m^{0,i})\}$$

   P_1 then sends $h(A_1), \ldots, h(A_t)$ and $h(B_1), \ldots, h(B_t)$ to P_2.
7. P_2 chooses a random index $e \in_R \{0,1\}^{\log(t)}$ and sends it to P_1.[3]
8. Let $O = \{1 \ldots e-1, e+1 \ldots t\}$. P_1 sends to P_2, s_i and s'_i for every $i \in O$. P_2 Computes $h(GC_i) = h(Garble(G, s_i, C'))$ for every $i \in O$ and verifies that they are equal to what he received from P_1. He also computes $G(s'_i)$ to get the decommitment values for commitments in A_i and B_i for every $i \in O$. P_2 then uses the keys and decommitments to recompute $h(A_i)$ and $h(B_i)$ on his own for every $i \in O$, and to verify that they are equal to what he received from P_1. If not, it outputs $\mathsf{corrupted}_1$ and halts.
9. P_1 sends to P_2 the actual garbled circuit GC_e, and the sets of commitment pairs A_e and B_e (note that P_2 only held $h(GC_e)$, $h(A_e)$, and $h(B_e)$). P_1 also sends decommitments to the input keys associated with his input for the circuit.
10. P_2 checks that the values received are valid decommitments to the commitments in B_e (he can open one commitment in every pair) and outputs $\mathsf{corrupted}_1$ if this is not the case.
11. Parties perform steps 5 and 6 of the OT protocols (see the full version of this paper for details regarding how this is done). P_1's input to the OTs are random strings corresponding to the eth circuit. As a result, P_2 learns one of the two strings $(k_{i+m}^{0,e}||r_{i+m}^{1,e}, k_{i+m}^{1,e}||r_{i+m}^{1,e})$ for the i^{th} OT $(1 \leq i \leq sm)$.
12. P_2 learns the decommitments and key values for his input bits from the OTs' outputs. He checks that the decommitments are valid for the commitments in A_e and that he received keys corresponding to his correct inputs. He outputs $\mathsf{corrupted}_1$ if this is not the case. He then proceeds with computing the garbled circuit $C'(x_1, z_2) = C(x_1, x_2)$, and outputs the result. If the keys are not correct and therefore he cannot compute the circuit, he outputs $\mathsf{corrupted}_1$.
13. If at anytime during the protocol one of the parties aborts unexpectedly, the other party will output abort and halt.

The general structure of our proof of security is the same as the proof in [AL07]. Due to lack of space details of the simulation are given in the full version of this paper. The following claim summarizes our result.

[3] For simplicity we assume that t is a power of 2.

Claim. Assuming that h is a collision-resistant hash function, Com_b is a perfectly binding commitment scheme, and G is a pseudorandom generator, then the above protocol is secure against covert adversaries with deterrence value $1 - 1/t$. The protocol runs in a constant number of rounds, and requires $O(|C| + sm + t)$ communication between the two players.

3.2 Extension to General Secure Two Party Computation

Our technique of only sending a hash (using a collision resistant hash function) of circuits and commitments directly generalizes to to the case of secure two party computation in the standard malicious adversary model.

Almost all the existing works for defending Yao's garbled circuit protocol against malicious adversaries in an efficient way [MF06,LP07,Woo07] use the *cut-and-choose* techniques. More specifically, party P_1 sends t garbled circuits to P_2; half of the circuits are chosen at random and their secrets are revealed by P_1; the remaining circuits are evaluated and the majority value is the final output of the protocol. Additional mechanisms are used to verify input consistency and to force the parties to use the same input values for majority of the circuits. Using our new garbling method and sending hash of circuits instead of the circuits themselves (as discussed previously) we automatically improve efficiency of these protocols. By carefully choosing the number of hashed garbled circuits and the fraction of circuits that are opened, we can make the efficiency gain quite substantial. Please see the full version of this paper for more detail on good choices of parameters. Next we outline some of these efficiency gains through some concrete examples.

Efficiency in Practice. For simplicity we demonstrate our improvements via comparison with the equality-checker scheme of [MF06] since a detailed analysis for it is available in [Woo07]. But, it is important to note that our techniques lead to similar improvements to all of the most-efficient protocols in the literature such as the expander-checker scheme of [Woo07] and the scheme proposed in [LP07] which also provides simulation-based security. Details of the modifications to the original equality-checker scheme are given in the full version of this paper.

By setting the parameters of the protocol (as we show in the full version of this paper), we can make the modified equality-checker (equality-checker-2) superior to the original one (equality-checker-1) in practice. The optimal choice of parameters depends on several factors such as the circuit size, the input size, and the size of the output of hash function. We work out some of these numbers in the full version to highlight the efficiency gained by using our techniques. Consider the following examples where the circuit are taken from [MNPS04]. Using those numbers, for a circuit that compares two 32-bit integers using 256 gates, our protocols roughly lead to factor of 12 improvement in communication complexity for the same probability of undetected cheating, and for a circuit that computes the median of two sorted arrays of ten 16-bit integers, with 4383 gates, we gain at least a factor of 30 improvement.

4 The Multi Party Case

We construct a multi party computation protocol secure against covert adversaries for a given deterrence parameter $1 - \frac{1}{t}$. Let there be n parties denoted by $P_1, \ldots, P_n$. The basic idea of the protocol is as follows. The parties run t parallel sessions, each session leading to the distributed generation of one garbled circuit. These sessions in the protocol are called the "garbled circuit generation sessions" (or GCG sessions in short). The protocol employed to generate these garbled circuits in the GCG sessions is a protocol secure only against semi honest adversaries and is based on the constant round BMR construction [BMR90]. Instead of employing zero knowledge proofs to go from semi-honest security to malicious security, we employ cut and choose techniques where the parties ensure the honesty of each other in $t - 1$ random GCG sessions. This is done by generating a shared challenge string which is used to select the one GCG session whose garbled circuit will be used for actual computation. The parties are required to reveal the (already committed) randomness used for every other GCG session. For a party, given the randomness and the incoming messages, the outgoing messages become deterministic. Hence the whole transcript of a GCG session can be checked (given randomness used by all the parties in this session) and any deviations can be detected.

The main problem which we face to turn this basic idea into a construction is that the secret inputs of the honest parties might be leaked since an adversarial party might deviate arbitrarily from the protocol in any GCG session (and this deviation is not detected until all the sessions have finished). This is because the distributed garbled circuit generation ideas in the BMR construction [BMR90] make use of the actual inputs of the honest parties (so that for each input wire, parties have the appropriate key required to evaluate the resulting garbled circuit). To solve this problem, we modify the BMR construction "from the inside" to enable these GCG sessions execute without using the inputs of the parties. Our modifications also allow the parties to check honesty of each other in these sessions without revealing their individual inputs (while still allowing the simulator to be able to extract these inputs during the proof of security).

4.1 Building Blocks

One of the building blocks of our protocol is a secure function evaluation protocol which is secure against honest-but-curious adversaries, and whose round complexity is proportional to the multiplicative depth of the circuit being evaluated (over $\mathbb{Z}_2 = GF(2)$). A textbook protocol such as that given by Goldreich [Gol04] (which is a variant of the semi-honest GMW protocol [GMW87]) suffices. We remark that this protocol will be used only to evaluate very short and simple circuits (such as computing XOR of a few strings).

We also need several subprotocols which are secure against standard (not only covert) malicious adversaries. We summarize these here:

- **Simulatable Coin Flipping From Scratch** (CoinFlipPublic)**:**
  This protocol emulates the usual coin-flipping functionality [Lin01] in the presence of arbitrary malicious adversaries. In particular, a simulator who controls a single player can control the outcome of the coin flip.

The remaining primitives assume the availability of a common random string σ. We assume that these primitives implement the corresponding ideal functionality in the CRS model.

- **Simultaneous commitment** ($\mathsf{Commit}_\sigma(x_1, ..., x_n)$)**:** Every player chooses a value x_i and commits to it. At the end of the protocol, the vector of commitments is known to all parties. The commitments are such that a simulator having trapdoor information about the CRS σ can extract the committed values.
- **Open commitments** ($\mathsf{OpenCom}_\sigma$)**:** Players simultaneously open their commitments over the broadcast channel.
  For the simulation to work, this protocol needs to be simulation-sound, in the following sense: if the simulator is controlling a subset of cheating players P_i, $i \in I_{sim}$, then he should be able to output a valid simulation in which all honest players lie about their committed values yet all cheating players are constrained to tell the truth or be caught.
- **Committed Coin Flipping** ($\mathsf{CommitedCoinFlipPublic}_\sigma$ **and** $\mathsf{CommittedCoinFlip}_\sigma\mathsf{To}P_i$)**:**
  Generates a commitment to a random string such that all players are committed to shares of the coin. In the second variant, P_i learns the random string and is committed to it.
- **Open coin:**
  Opens a committed coin to all players over the broadcast channel. The simulator should be able to control the coin flip.

These primitives can be implemented very efficiently under several number-theoretic assumptions. For concreteness, we have described efficient instantiations based on the DDH assumption in the full version of this paper. These are summarized here.

Lemma 1. *Suppose the Decisional Diffie-Hellman problem is hard in group G. There exist secure implementations of the protocols above. The CRS protocols (*Commit_σ*,* $\mathsf{OpenCom}_\sigma$*,* $\mathsf{CommitedCoinFlipPublic}_\sigma$*,* $\mathsf{CommittedCoinFlip}_\sigma\mathsf{To}P_i$*) require $O(n\ell + n^2k)$ bits of communication each, and a shared CRS of length $2n+1$ group elements. Here k is the bit length of the elements of the group G, and ℓ is the bit length of the strings being generated, committed, or opened. Generating a CRS of length ℓ bits via* CoinFlipPublic *requires $O(n^2 \log(n)k + n\ell)$ bits of communication and $O(\log n)$ rounds.*

4.2 Main Multiparty Protocol

We now turn to the protocol itself. Let C be a circuit corresponding to the function $f(x_1, x_2, \ldots, x_n)$ which the parties wish to jointly compute. We denote the total number of wires (including the input and output wires) in C by W, each having index in the range 1 to W . Let F and G be pseudorandom generators with seed length s (here s is the security parameter). The parties run the following protocol.

Stage 0. Collectively flip a single string σ having length $poly(s)$. The string σ is used as a CRS for the commitment and coin-flipping in the remaining stages of the protocol.

$$\sigma \leftarrow \mathsf{CoinFlipPublic}$$

Stage 1. The parties generate the commitment to a shared challenge random string $e \in [t]$

$$e \leftarrow \mathsf{CommitedCoinFlipPublic}_\sigma$$

The challenge e will later be used to select which of the GCG sessions (out of the t sessions) will be used for actual computation. The parties will be required to show that they were honest in all other GCG sessions (by revealing their randomness).

Stage 2. For each $i \in [n]$ and $S \in [t]$, collectively flip coins $r_i[S]$ of length s and open the commitment (and decommitment strings) to P_i only:

$$r_i[S] \leftarrow \mathsf{CommittedCoinFlip}_\sigma\mathsf{To}P_i$$

Thus, a party P_i obtains a random string $r_i[S]$ for every session $S \in [t]$. All other parties have obtained commitment to $r_i[S]$. The random string $r_i[S]$ can be expanded using the pseudorandom generator F. It will be used by P_i for the following:

- To generate the share $\lambda_i^w[S] \in \{0,1\}$ of the *wire mask* $\lambda^w[S]$ (in Stage 3 of our protocol) for every wire w in the garbled circuit $GC[S]$ to be generated in session S. Recall that in a garbled circuit $GC[S]$, for every wire w, we have two *wire keys* (denoted by $k^{w,0}[S]$ and $k^{w,1}[S]$): one corresponding to the bit on wire w being 0 and the other to bit being 1 (during the actual evaluation of the garbled circuit, a party would only be able to find one of these keys for every wire). The wire mask determine the correspondence between the two wire keys and the bit value, i.e., the key $k^{w,b}[S]$ corresponds to the bit $b \oplus \lambda^w[S]$.
- To run the GCG session S (i.e., Stage 4 of our protocol). Note that we generate the wire masks for the garbled circuits in stage 3 (instead of 4) to enable the parties to run stage 4 without using their inputs.

Stage 3. Every player P_i is responsible for a subset of the input wires J_i, and holds an input bit x^w for each $w \in J_i$. For every $w \in J_i$, and session S, P_i computes $I^w[S] = x^w \oplus \lambda_i^w[S]$. For each S, players simultaneously commit to the value I^w for each of their input wires (each input wire is committed to by exactly one player):

$$\{COM(I^w[S]) \ : \ \text{input wires} \ \ w\} \leftarrow \mathsf{Commit}_\sigma\Big(\{I^w[S] \ : \ S \in \{1,...,t\}, \ w \in \text{input wires } \}\Big)$$

Recall that exactly one of the sessions will be used for actual secure function evaluation. In that session, the above commitment will be opened and $x^w \oplus \lambda_i^w[S]$ will be revealed (however $\lambda_i^w[S]$ will remain hidden). In rest of sessions where the garbled circuit generated will be opened and checked completely by all the parties, the wire mask share $\lambda_i^w[S]$ will be revealed (since its a part of the garbled circuit description and generated using randomness $r_i[S]$). However the above commitment to $x^w \oplus \lambda_i^w[S]$ will *not* be opened for those sessions. This ensures the secrecy of the input x^w (while still allowing to simulator to extract it in our proof of security).

Stage 4. This is the stage in which the parties run t parallel garbled circuit generation session. This stage is based on the BMR construction but does not make use of the inputs of the parties. Each session in this stage can be seen as an independent efficient protocol (secure against honest but curious adversaries) where:

- In the beginning, the parties already hold shares of the wire masks $\lambda_i^w[S]$ to be used for the garbled circuit generation (as opposed to generating these wire masks in this protocol itself).
- In the end, the parties hold a garbled circuit $GC[S]$ for evaluating the function f. Furthermore, each party also holds parts of the wire keys for input wires (such that when for all input wires, all the parts of the appropriate wire key are broadcast, the parties can evaluate the garbled circuit; which key is broadcast is decided by the openings of the commitments of stage 3).

We now describe this stage in more detail.

1. P_i broadcasts the wire mask shares $\lambda_i^w[S]$ for all input wires belonging to other players (i.e., for w not in J_i), and for all output wires. Thus only the masks for P_i's inputs, and for internal wires, remain secret from the outside world . Note that $\lambda^w[S] = \bigoplus_{i=1}^n \lambda_i^w[S]$ is the wire mask for wire w. Each player holds shares of the wire masks.
2. For every wire w of the circuit C, P_i generates two random *key parts* $k_i^{w,0}[S]$ and $k_i^{w,1}[S]$. The full wire keys are defined as the concatenation of the individual key parts. That is, $k^{w,0}[S] = k_1^{w,0}[S] \circ \ldots \circ k_n^{w,0}[S]$ and $k^{w,1}[S] = k_1^{w,1}[S] \circ \ldots \circ k_n^{w,1}[S]$.
3. Recall that for every gate in the circuit, the wire keys of incoming wires will be used to encrypt the wire keys for outgoing wires (to construct what is called a *gate table*). However it is not desirable to use a regular symmetric key encryption algorithm for this purpose. The reason is that the gate tables will be generated by using a (honest but curious) secure function evaluation protocol (see next step) and the complexity of the circuit to be evaluated will depend upon the complexity of the encryption algorithm. To avoid this problem, the parties locally expand their key parts into large strings (and then later simply use a one time pad to encrypt). More precisely, P_i expands the key parts $k_i^{w,0}[S]$ and $k_i^{w,1}[S]$ using the pseudorandom generator G to obtain two new keys, i.e.,$(p_i^{w,\ell}[S], q_i^{w,\ell}[S]) = G(k_i^{w,\ell}[S])$, for $\ell \in \{0,1\}$. Each of the new keys has length $n|k_i^{w,\ell}[S]|$ (enough to encrypt a full wire key).

4. The players then run a Secure Function Evaluation protocol secure against *honest-but-curious* adversaries to evaluate a simple circuit to generate the *gate tables*. This stage is inspired by a similar stage of the Beaver et al. protocol [BMR90]. This is the step that dominates the computation and communication complexity of our construction. However as opposed to BMR, *the underlying multi-party computation protocol used here only needs to be secure against semi-honest adversaries.* More details follow.

   For every gate g in the circuit C, define a gate table as follows. Let a, b be the two input wires and c be the output wire for the gate g, and denote the operation performed by the gate g by $\otimes$ (e.g. AND, OR, NAND, etc). Before the protocol starts, P_i holds the following inputs: $p_i^{a,\ell}[S], q_i^{a,\ell}[S], p_i^{b,\ell}[S], q_i^{b,\ell}[S], k_i^{c,\ell}[S]$ where $\ell \in \{0,1\}$ along with shares $\lambda_i^a[S], \lambda_i^b[S], \lambda_i^c[S]$ of wire masks $\lambda^a[S], \lambda^b[S], \lambda^c[S]$. P_i runs the protocol along with other parties to compute the following gate table:

$$
\begin{aligned}
A_g &= p_1^{a,0}[S] \oplus \ldots \oplus p_n^{a,0}[S] \oplus p_1^{b,0}[S] \oplus \ldots \oplus p_n^{b,0}[S] \\
&\quad \oplus \begin{cases} k_1^{c,0}[S] \circ \ldots \circ k_n^{c,0}[S] & \text{if } \lambda^a[S] \otimes \lambda^b[S] = \lambda^c[S] \\ k_1^{c,1}[S] \circ \ldots \circ k_n^{c,1}[S] & \text{otherwise} \end{cases} \\
B_g &= q_1^{a,0}[S] \oplus \ldots \oplus q_n^{a,0}[S] \oplus p_1^{b,1}[S] \oplus \ldots \oplus p_n^{b,1}[S] \\
&\quad \oplus \begin{cases} k_1^{c,0}[S] \circ \ldots \circ k_n^{c,0}[S] & \text{if } \lambda^a[S] \otimes \overline{\lambda^b[S]} = \lambda^c[S] \\ k_1^{c,1}[S] \circ \ldots \circ k_n^{c,1}[S] & \text{otherwise} \end{cases} \\
C_g &= p_1^{a,1}[S] \oplus \ldots \oplus p_n^{a,1}[S] \oplus q_1^{b,0}[S] \oplus \ldots \oplus q_n^{b,0}[S] \\
&\quad \oplus \begin{cases} k_1^{c,0}[S] \circ \ldots \circ k_n^{c,0}[S] & \text{if } \overline{\lambda^a[S]} \otimes \lambda^b[S] = \lambda^c[S] \\ k_1^{c,1}[S] \circ \ldots \circ k_n^{c,1}[S] & \text{otherwise} \end{cases} \\
D_g &= q_1^{a,1}[S] \oplus \ldots \oplus q_n^{a,1}[S] \oplus q_1^{b,1}[S] \oplus \ldots \oplus q_n^{b,1}[S] \\
&\quad \oplus \begin{cases} k_1^{c,0}[S] \circ \ldots \circ k_n^{c,0}[S] & \text{if } \overline{\lambda^a[S]} \otimes \overline{\lambda^b[S]} = \lambda^c[S] \\ k_1^{c,1}[S] \circ \ldots \circ k_n^{c,1}[S] & \text{otherwise} \end{cases}
\end{aligned}
$$

This circuit has multiplicative depth 2. If we use the honest-but-curious SFE protocol from [Gol04], this stage requires a constant number of rounds.

At the end of this phase, for each session S, the parties hold a garbled circuit $GC[S]$ (which consists of the gate tables as generated above, along with the wire masks $\lambda^w[S]$ for each *output wire* w).

Stage 5. The parties now open the challenge e generated in Step 3, using $\mathsf{OpenCom}_\sigma$.

Stage 6. For each session $S \neq e$, each party P_i opens the commitment to $r_i[S]$ generated in Step 1. Given $r_1[S], \ldots, r_n[S]$, all the wire mask shares and the protocol of Stage 4.2 become completely deterministic. More precisely, each player can regenerate the transcript of Stage 4.2, and can thus verify that all parties played honestly for all sessions $S \neq e$. If P_i detects a deviation from the honest behavior, it aborts identifying the malicious party P_j who deviated.

Note that the only point so far where the parties were required to use their inputs is Stage 3 (where P_i committed to $x^w \oplus \lambda_i^w[S]$ for all $w \in J_i$). However these commitments were not used in any other stage. Hence, since these commitments have not yet been opened nor used anywhere else, if the players abort at this stage then no information is learned by the adversary.

Once the parties successfully get past this stage without aborting, we have a guarantee that the garbled circuit $GC[e]$ was correctly generated except with probability $\frac{1}{t}$. Thus, $\frac{1}{t}$ bounds the probability with which an adversary can cheat successfully in our protocol.

Stage 7. For all input wires $w \in J_i$, P_i now opens the commitments $COM^w[e]$ (see Stage 3) using $\mathsf{OpenCom}_\sigma$, thus revealing $I^w = \lambda_i^w[e] \oplus x^w$. Set $L^w = I^w \oplus \bigoplus_{j=1}^{i-1} \lambda_j^w[e] \oplus \bigoplus_{j=i+1}^{n} \lambda_j^w[e]$ (where $\lambda_j^w[e]$ was broadcast in stage 4(a)), i.e., $L^w = \lambda^w[e] \oplus x^w$. Every party P_ℓ, $1 \leq \ell \leq n$ broadcasts the key parts $k_\ell^{w,L^w}[e]$.

Stage 8. P_i now has the garbled circuit $GC[e]$ as well the wire keys $k^{w,L^w}[e] = k_1^{w,L^w}[e] \circ \ldots \circ k_n^{w,L^w}[e]$ for all input wires w of the circuit. Hence P_i can now evaluate the garbled circuit on its own in a standard manner to compute the desired function output $C(x_1, x_2, \ldots, x_n)$. For more details on how the garbled circuit $GC[e]$ is evaluated, see [BMR90].

The following theorem summarizes our result. See the full version of this paper for the analysis of our construction.

Theorem 2. *If the coin-flipping and commitment primitives are secure against malicious adversaries and the SFE scheme is secure against honest-but-curious adversaries, then the above construction is secure in the presence of covert adversaries with* $1 - \frac{1}{t}$ *deterrence.*

If we instantiate the coin-flippping and commitment primitives as in Lemma 1, and use the SFE scheme of [Gol04], then the protocol above requires $O(\log n)$ *rounds and a total of* $O(n^3 ts|C|)$ *bits of communication to evaluate a boolean circuit of size* $|C|$*, where* s *is the security parameter (the input size of a pseudorandom generator). The computational complexity is the same up to polylogarithmic factors.*

If we use the constant-round coin-flipping protocols of Katz et al. [KOS03] or Pass [Pas04], then the protocol above runs in constant rounds, but requires substantially slower (though still polynomial) computations.

The protocol above is the first multiparty protocol we know of which is tailored to covert adversaries. As a point of comparison, to our knowledge the most efficient protocol secure against *malicious* adversaries that tolerates up to $n-1$ cheaters is that of Katz et al. [KOS03]. The running time of the KOS protocol is dominated by the complexity of proving statements *about* circuits of size $O(n^3 s|C|)$ (this is the cost incurred by compiling an honest-but-curious SFE protocol). In contrast, our protocol runns in time $\tilde{O}(n^3 st)$. Thus, the contribution of this protocol can be seen as relating the complexity of security against covert adversaries to security against honest-but-curious adversaries:

$$\text{Cost of deterrence } 1 - \frac{1}{t} \text{ against covert adversaries}$$
$$t \cdot \Big(\text{Cost of honest-but-curious garbled circuit generation}\Big)$$

References

AL07. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (2007)

Bar02. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: FOCS, pp. 345–355. IEEE Computer Society Press, Los Alamitos (2002)

BMR90. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: STOC, pp. 503–513. ACM Press, New York (1990)

BOGW88. Ben-Or, M., Goldwasser, S., Widgerson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of ACM STOC, pp. 1–10 (1988)

CCD88. Chaum, D., Crepeau, C., Damgard, I.: Multi-party unconditionally secure protocols. In: Proceedings of ACM STOC, pp. 11–19 (1988)

CO99. Canetti, R., Ostrovsky, R.: Secure computation with honest-looking parties: What if nobody is truly honest (extended abstract). In: STOC, pp. 255–264 (1999)

CR87. Chor, B., Rabin, M.O.: Achieving independence in logarithmic number of rounds. In: PODC, pp. 260–268 (1987)

DI05. Damgård, I., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 378–394. Springer, Heidelberg (2005)

FY92. Franklin, M., Yung, M.: Communication complexity of secure computation (extended abstract), pp. 699–710 (1992)

GMW87. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: proceedings of 19th Annual ACM Symposium on Theory of Computing, pp. 218–229 (1987)

Gol04. Goldreich, O.: Foundation of Cryptography, Volume II: Basic Applications. Cambridge University Press, Cambridge (2004)

IKLP06. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Kleinberg, J.M. (ed.) STOC, pp. 99–108. ACM Press, New York (2006)

JS07. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, Springer, Heidelberg (2007)

KOS03. Katz, J., Ostrovsky, R., Smith, A.: Round efficiency of multi-party computation with a dishonest majority. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 578–595. Springer, Heidelberg (2003)

Lin01. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 171–189. Springer, Heidelberg (2001)

LP04. Lindell, Y., Pinkas, B.: A proof of yao's protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175 (2004)

LP07. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, Springer, Heidelberg (2007)

MF06. Mohassel, P., Franklin, M.: Efficiency tradeoffs for malicious two-party computation. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, Springer, Heidelberg (2007)

MNPS04. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system (2004)

Pas04. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: Babai, L. (ed.) STOC, pp. 232–241. ACM Press, New York (2004)

Pin03. Pinkas, B.: Fair secure two-party computation. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 87–105. Springer, Heidelberg (2003)

Woo07. Woodruff, D.P.: Revisiting the efficiency of malicious two-party computation. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, Springer, Heidelberg (2007)

Yao86. Yao, A.C.: How to generate and exchange secrets. In: Proceedings of the 27th IEEE symposium on Foundations of Computer science, pp. 162–167 (1986)

Almost-Everywhere Secure Computation

Juan A. Garay[1,⋆] and Rafail Ostrovsky[2,⋆⋆]

[1] Bell Labs, Alcatel-Lucent, 600 Mountain Ave., Murray Hill, NJ 07974
garay@research.bell-labs.com
[2] Departments of Computer Science and Mathematics, UCLA, Los Angeles, CA 90095-1596
rafail@cs.ucla.edu

Abstract. Secure multi-party computation (MPC) is a central problem in cryptography. Unfortunately, it is well known that MPC is possible if and only if the underlying communication network has very large connectivity — in fact, $\Omega(t)$, where t is the number of potential corruptions in the network. This impossibility result renders existing MPC results far less applicable in practice, since many deployed networks have in fact a very small degree.

In this paper, we show how to circumvent this impossibility result and achieve meaningful security guarantees for graphs with small degree (such as expander graphs and several other topologies). In fact, the notion we introduce, which we call *almost-everywhere MPC*, building on the notion of almost-everywhere agreement due to Dwork, Peleg, Pippenger and Upfal, allows the degree of the network to be much smaller than the total number of allowed corruptions. In essence, our definition allows the adversary to *implicitly* wiretap some of the good nodes by corrupting sufficiently many nodes in the "neighborhood" of those nodes. We show protocols that satisfy our new definition, retaining both correctness and privacy for most nodes despite small connectivity, no matter how the adversary chooses his corruptions.

Instrumental in our constructions is a new model and protocol for the *secure message transmission* (SMT) problem, which we call *SMT by public discussion*, and which we use for the establishment of pairwise secure channels in limited connectivity networks.

Keywords: Secure multi-party computation, secure message transmission, almost-everywhere agreement, expander graphs, bounded-degree networks.

1 Introduction

Secure multi-party computation (MPC) [33,21,2,6] is one of the most fundamental problems in cryptography. Simply put, in MPC n players jointly compute and obtain the value of an arbitrary n-ary polynomial-time computable function on their inputs, in such way that even if some fraction of the players are corrupted by a malicious adversary, the correct outputs as well as the privacy of the inputs of the uncorrupted (honest) players are

⋆ A portion of this work was done while the authors were visiting the Institute for Pure and Applied Mathematics (IPAM), UCLA.

⋆⋆ Supported in part by an IBM Faculty Award, a Xerox Innovation Group Award, NSF grants no. 0430254, 0716835 and 0716389, and a MICRO grant from the University of California.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 307–323, 2008.

guaranteed. After its formulation, MPC has been studied extensively, and many flavors with regards to, for example, the type of corruptions allowed by the adversary, its computational power, and definitions of security, have been considered in the literature.

In this paper our focus is on unconditional, or information-theoretic, secure multiparty computation, as considered in [2,6], where no restrictions are placed on the computational power of the adversary. In this setting, assuming that players can communicate with every other player over not only dedicated but also private communication channels, MPC is achievable as long as more than $\frac{2}{3}$ of the players remain uncorrupted. Moreover, this bound on the number of players is tight.

The above scenario, however, assumes point-to-point private communication channel between *every pair of players*. In fact, with a few noted exceptions (more on this below), this is not only true for unconditional MPC, but for most of the work on other models as well. Since reliable, let alone private communication is costly to achieve, fully connected networks sound like a prohibitive proposition. Indeed, this question was posed by Dolev [9], and later by Dolev, Dwork, Waarts and Yung [10], whose combined results show that in fact if there are t corrupted players, then $(2t+1)$-connectivity is both necessary and sufficient for unconditional MPC. On the other hand, typically in practical networks most nodes have a very small connectivity, which is independent of the size of the network.

In this paper, we show that meaningful statements about unconditional MPC can be made even if every node has small, even constant, connectivity. Clearly, in such a setting, we must give up on some of the honest nodes, for example, in the case of such nodes being totally "surrounded" in the network by corrupted nodes; still, we would like to be able to guarantee the security of a large fraction of uncorrupted nodes.

Our notion of "giving up" nodes originates from the notion of *almost-everywhere agreement* proposed by Dwork, Peleg, Pippenger and Upfal [11], who study how to achieve Byzantine agreement [26,25] in a limited connectivity setting. We build on the notions developed in [11], and in some sense, our central result can be viewed as a generalization of theirs. In essence, given a broadcast channel (implied by Byzantine agreement) and a constant number of uncorrupted paths among a large subset of nodes in the network, we show how to implement secure (i.e., reliable and private) channels among them, and thus achieve *almost-everywhere MPC*. "In essence" because apart from the construction, as opposed to just achieving the correctness property for a particular MPC instance (Byzantine agreement), substantial additional efforts are required in order to capture the privacy requirement of general MPC, particularly in the case of adaptive corruptions, and proving it correct.

One of the tools that we use to achieve the above transformation, which we call *secure message transmission by public discussion*, might be of independent of interest. As mentioned above, Dolev *et al.* [10] show a tight $(2t+1)$-connectivity bound for the (perfectly) secure message transmission (SMT) problem, of one node sending a message perfectly privately and correctly to another node over a network. That many channels, in fact, are needed to establish a public channel (no privacy concerns, only reliability). In our model, a public channel for large subsets of nodes can be constructed in a different way, and thus we are able to let the adversary corrupt *all but one* of the channels connecting those nodes, at the expense of a small error.

We stress that while in almost-everywhere MPC we give up the privacy and correctness of some of the nodes, we consider this a realistic assumption. Indeed, the requirements on a corrupted node are also given up in the classical model of MPC, and what we define here is a model where the adversary by corrupting some t nodes, can potentially corrupt another set of t' nodes, which depends on t and the particular network. However, as long as the entire set is sufficiently small, i.e., $t + t' < \frac{n}{3}$, it is guaranteed that the rest of the nodes can achieve the requirements of secure multi-party computation, even on networks with bounded degree.

Related work. As already mentioned, our new notion is closely related to the concept of almost-everywhere agreement introduced in [11], and further explored in [30,5]. We review the results in [30] in Section 4.3, where we describe especial networks where almost-everywhere MPC can be achieved.

We also already mentioned the relation of our tool for secure channels to the problem of *perfectly secure message transmission* [10], which has been further studied in, e.g., [28,8,29,1,12,24]. In [15], Franklin and Wright take a different approach and study the necessary and sufficient conditions for secure message transmissions over multicast lines.

Our model for secure message transmission by public discussion is also related to the one used in *privacy amplification* and *secret key agreement* ([4,3] and extensive number of follow-ups), where there also is an authentic public channel, and a private channel which the adversary is allowed to eavesdrop and/or tamper, depending on the various sub-models. Our problem can be viewed as a special instance of secret key agreement in the presence of severe tampering and transmission errors, but for a very specialized tampering function, if we view all the channels as a combined channel.

"Hybrid" failure models have been considered in the literature, where the adversary is allowed to maliciously corrupt some fraction of the players and in addition to cause some more benign form of failure to some other players [19,14]. In [14] in particular, Fitzi, Hirt and Maurer allow the adversary to eavesdrop on the additional players. In our model, the potential additional eavesdropping (as well as violation of correctness) is defined structurally, given the graph topology and the location of the truly corrupted nodes.

Finally, the problem statement of almost-everywhere secure computation, as well as the overall approach are joint work with Shailesh Vaya [31,17,18]. (See Acknowledgements for a more detailed account on this collaboration.) The work reported in [32] follows this approach as well: adding privacy to networks that admit almost-everywhere agreement, using a protocol cast there as achieving secret key agreement by public discussion (see above). Besides several other issues, a salient difference (shortcoming) in [32] is the approach to a simulation-based security definition [22], where the ideal-world adversary (the simulator), besides having access to the inputs of the corrupted players, is also given access to the inputs of the honest players that are given up; such a strong assumption gives the simulator an additional unfair advantage compared to the real-world adversary. In contrast, in this paper we propose an indistinguishability-based security definition, known to be weaker than simulation-based, but meaningful. Further remarks on definitional issues are included in Sections 2.2 and 5.

Organization of the rest of the paper and our contributions. The rest of the paper is organized as follows. In Section 2 we present the model for and our definition of almost-everywhere MPC, together with other building blocks that will be used in our construction. In Section 3 we present the new model and protocol for SMT by public discussion, which we then use to obtain secure channels. Section 4 is dedicated to almost-everywhere MPC. First (Section 4.1), we define a class of graphs with special properties, which we call *almost-everywhere admissible graphs*. The literature on almost-everywhere agreement describes several such graphs; however, not all of them are suited to satisfy the privacy requirements of our application. We show an efficient transformation of graphs G_n with degree d that allow almost-everywhere agreement into a graph $G_{2n}$of degree $O(d)$ that is almost-everywhere admissible. We then show (Section 4.2) how to construct protocols for almost-everywhere MPC on such graphs satisfying our definition, followed by concrete results for some specific networks (Section 4.3). We conclude in Section 5 with a summary and directions for future work.

2 Model, Definitions and Tools

In this paper we consider networks (graphs) $G = (V, E)$ that are *not* fully connected, as in [11,30,5]. We let $|V| = n$. We will also refer to the nodes in V as "players," and to the edges in E as (communication) links or channels. The networks are synchronous, and the computation can be divided into rounds; in each round, a player may send a (possibly different) message on each of its incident links, and messages sent in one round are delivered before the next round. Up to t of the players can be actively corrupted by an adversary $\mathcal{A}$; we will use $T \subset V$, $|T| = t$ to denote the set of corrupted players, and sometimes we will refer to $\mathcal{A}$ as a t-adversary. We assume that $\mathcal{A}$ has unlimited computational power, and, furthermore, that $\mathcal{A}$ is *rushing*, meaning he can learn the messages sent by the uncorrupted players in each round before deciding on the messages of corrupted players for this round, and *adaptive*, meaning that information obtained from a set of corrupted players at a particular round can affect the choice of the next player(s) to be corrupted.

2.1 Building Blocks

Our protocols for almost-everywhere MPC will be using several building blocks, including *almost-everywhere agreement* [11], *verifiable secret sharing* (VSS) [7], and a new primitive introduced in this paper that we call *secure message transmission by public discussion* (Section 3).

Byzantine agreement and almost-everywhere agreement. We start with the standard definition of Byzantine agreement [25,26]. Here the network model is that of a fully connected network of pairwise authenticated channels.

Definition 1. *A protocol for parties* $\{P_1, \ldots, P_n\}$*, each holding an initial value* v_i*, is a* Byzantine agreement protocol *if the following conditions hold for any* t*-adversary:*

- AGREEMENT: *All honest parties output the same value.*
- VALIDITY: *If for all honest parties* $v_i = v$*, then all honest parties output* v*.* ◇

It is known that $n > 3t$ is necessary and sufficient for Byzantine agreement [25,26], and there exist efficient (polynomial-time and round-optimal) deterministic protocols achieving it [16]. We will in fact rely on the related task called *broadcast*, where there is a distinguished player (the *sender*) P^* holding an initial value v. The agreement condition remains the same as above; validity requires that if the sender is honest, then all honest players output v. Broadcast easily reduces to Byzantine agreement, preserving the above bound on the number of players (in the information-theoretic setting).

In [11], Dwork, Peleg, Pippenger and Upfal relax the full (more specifically, $\Omega(t)$) connectivity requirement of the original Byzantine agreement formulation, proposing *almost-everywhere agreement* — "almost everywhere" because with partial connectivity agreement involving all the honest players is not possible and one must settle for agreement with *exceptions*, where some of the honest players are left out. Thus, in this context, the number of exceptions constitutes another relevant parameter for agreement and broadcast protocols. Dwork *et al.* consider several classes of networks depending on their degree; the general approach, however, is to show how to simulate the sending of a message from one player to another in the fully connected setting by a transmission scheme working over multiple paths on the partially connected network in such a way that if none of the players belongs to a set, call it T^+, which includes the set of corrupted players (T) plus the left-out honest players, then the simulation is faithful. In turn, this makes it possible to simulate any Byzantine agreement protocol for fully connected networks which does not rely on the privacy of the links by treating players from T^+ as corrupted. We further review and apply some of the results in [11], as well as those in follow-up work [30], in Section 4.3.

As before in the full connectivity case, an *almost-everywhere broadcast* protocol can be derived by having the sender first (attempt to) send his value to all other players using the transmission simulation scheme, and then having all players run the almost-everywhere agreement protocol; for an honest sender in T^+ the validity condition is not guaranteed, but agreement guarantees that all the players in $V - T^+$ will output the same value.

Verifiable secret sharing. Here the network model is that of a fully connected network of pairwise secure channels. One of the players is given a special role of being the *dealer* D. A VSS protocol consists of two phases: in the first phase, the dealer D distributes a secret s, while in the second, taking place possibly at a later time, the players cooperate in order to retrieve it. A more detailed specification is as follows:

Sharing phase: The dealer initially holds secret $s \in K$ where K is a finite field of sufficient size; at the end of the phase each player P_i holds some private information v_i.

Reconstruction phase: Each player P_i reveals his private information v_i. Then, on the revealed information v'_i (a corrupted player may reveal $v'_i \neq v_i$), a reconstruction function is applied in order to compute the secret, i.e., $s = Rec(v'_1, \ldots, v'_n)$.

The guarantees that are required from a VSS protocol are as follows.

Definition 2. *An n-player protocol is called a* (perfect) (n, t)-VSS protocol *if, for any t-adversary, the following condition holds:*

- PRIVACY: *If D is honest, then the adversary's view during the sharing phase reveals no information about s. More formally, the adversary's view is identically distributed under all different values of s.*
- CORRECTNESS: *If D is honest, then the reconstructed value is equal to the secret s.*
- COMMITMENT*: After the sharing phase, a unique value s^* is determined which will be reconstructed in the reconstruction phase; i.e., $s^* = Rec(v'_1, \cdots, v'_n)$ regardless of the information provided by the corrupted players.* $\diamond$

It is known that $n > 3t$ is necessary and sufficient for VSS [2], and there exist efficient protocols achieving it [20,13][1]. If an (negligible) error is allowed, and additionally a broadcast channel is given, then $n > 2t$ suffices [27].

The last tool that we will be using, *secure message transmission by public discussion*, we treat separately in Section 3.

2.2 Almost-Everywhere MPC

We now turn to the formulation of *almost-everywhere secure multi-party computation*. It follows from results in [9,10] that in the type of networks that we are considering, it is not possible to establish secure channels between every pair of nodes, a known requirement for MPC. Indeed, depending on connectivity patterns, some nodes in V may have a majority (or even all) of the links coming from nodes controlled by $\mathcal{A}$. Thus, and as in [11,30,5] in the context of almost-everywhere agreement, our approach to secure multi-party computation on such networks is also to "give up" on those nodes.

More formally, let $\mathcal{V}$ denote the power set of V, $\mathcal{V}^{(\leq t)}$ the set of all subsets of V of size at most t, and let $\mathcal{X} : \mathcal{V}^{(\leq t)} \to \mathcal{V}$ be a function with the following properties:

1. $\mathcal{X}$ is monotically increasing, i.e., $T_1 \subset T_2$, implies $\mathcal{X}(T_1) \subset \mathcal{X}(T_2)$; and
2. $T \subset \mathcal{X}(T)$.

We say a protocol Π achieves X *secure multi-party computation* (X-MPC for short), where $X \stackrel{\text{def}}{=} \max_{T \subset V, |T|=t}\{|\mathcal{X}(T)|\}$, if for every subset T of nodes controlled by the t-adversary by the end of the protocol, there exists a set $W \subset V$ of uncorrupted players, $|W| \geq n - X$, such that all the players in W are able to perform secure multi-party computation. In the case of a fully connected network, $\mathcal{X}(T) = T$. Sometimes we will refer to the players in W as *privileged*, and to the players in $\mathcal{X}(T) - T$ as *doomed*.

Recall that the two main requirements in MPC are correctness of the output of the function being computed and privacy of the honest players' inputs. Prior work mentioned above for the limited connectivity setting was only concerned with the correctness of a function; given the additional privacy requirement of MPC, specifying what "to able to perform secure multi-party computation" means becomes more challenging. This gets further complicated by the fact that we are considering adaptive adversaries, which implies that the sets defined above might change (in particular, the set of given-up players will grow) during the execution of the protocol, and we would like, for any protocol, to state security guarantees for the honest players as these sets change.

[1] In fact, these protocols additionally assume the availability of a broadcast channel, which can be implemented on the fully connected point-to-point network, since $n > 3t$.

The "commit-and-compute" paradigm. Typically, MPC protocols to compute a function on the inputs of the players $f(x_1, x_2, \cdots, x_n)$ (assuming for simplicity that all the players get the same result) tolerating active adversaries would start with the players executing a commitment phase, where the players' inputs are shared among the rest of the players, followed by a computation phase, followed by an output phase. For X-MPC, we make the commitment phase explicit and part of the definition, as this will allow us to precisely state the conditions on nodes in an unfavorable connectivity situation.

Definition 3. *Let $G_n = (V, E)$, $|V| = n$ be a network, and T, $\mathcal{X}(T)$, X and W as defined above. An n-player two-phase protocol is an X* secure multi-party computation protocol *if for any probabilistic polynomial-time computable function f, the following two conditions are satisfied at the end of the respective phases:*

Commitment phase: *During this phase, all players in V commit to their inputs.*

- BINDING: *For all $P_i \in V$, there is a uniquely defined value x_i^*; if $P_i \in W$, then $x_i^* = x_i$.*
- PRIVACY: *For all players $P_i \in W$, x_i^* is information-theoretically hidden.*

Computation phase:

- CORRECTNESS: *For all players $P_i \in W$, $f(x_1^*, x_2^*, \cdots, x_n^*)$ is the value output by P_i.*
- PRIVACY: *Consider two runs of the protocol such that $\mathcal{X}(T_1) = \mathcal{X}(T_2)$ (and thus $W_1 = W_2 = W$) and let $\overrightarrow{x_S^*}$ denote the vector of committed inputs corresponding to players in a given set S. If for all $\overrightarrow{x_W^*}$, $\overrightarrow{z_{\mathcal{X}(T_1)}^*}$, $\overrightarrow{y_W^*}$, $\overrightarrow{z_{\mathcal{X}(T_2)}^*}$ it holds that $f(\overrightarrow{x_W^*}, \overrightarrow{z_{\mathcal{X}(T_1)}^*}) = f(\overrightarrow{y_W^*}, \overrightarrow{z_{\mathcal{X}(T_2)}^*})$, then the adversary's views in the two runs are statistically indistinguishable.* ◇

We now make some remarks regarding our X-MPC definition.

Remark 1. In the adaptive-adversary setting, the sets T, $\mathcal{X}(T)$ and W might change dynamically during the execution of a protocol. Thus, we stress that in the definition above these sets are always defined with respect to the completion of a phase.

Remark 2. It is well known that an information-theoretic definition of privacy in terms of indistinguishability is weaker than a simulation-based counterpart. For example, consider a secure — according to our definition — multi-party protocol to compute $f(x)$ for a one-way permutation f, where x should remain hidden from all players. Information theoretically, the computation of $f(x)$ and the computation that would reveal x reveals the same amount of information to an infinitely powerful adversary; however, in the latter case, clearly x does not remain hidden. This example, due to Canetti, illustrates that one should not "mix" information-theoretic notions and computational notions, and that only suitable properties, such as those guaranteed by information-theoretically secure MPC protocols ([2,6] and follow-ups), will remain secure according to our definition. See Section 5 for further remarks on simulation-based definitions for the X-MPC setting.

Before turning to protocols for X-MPC, in the next section we introduce the last tool that our protocols will be using, which will allow for the establishment of secure channels in the limited connectivity setting.

3 Secure Message Transmission by Public Discussion

Let us first specify the (new) communication model that we are considering in this section; we will then relate this model to the X-MPC context. Here we consider just two players, $\mathcal{S}$ and $\mathcal{R}$, connected by a set of channels $\mathcal{C} = \{C_1, ..., C_N\}$, the contents of all but one of which can be eavesdropped and modified (in an arbitrary manner) by an adaptive, computationally unbounded adversary $\mathcal{A}$. Additionally, $\mathcal{S}$ and $\mathcal{R}$ have at their disposal an authentic and reliable public channel *Pub*.

The goal is to realize, using this communication model, a means for $\mathcal{S}$ to securely send messages to $\mathcal{R}$, a functionality known as *secure message transmission* (SMT) [10]. We will later be using this version of SMT in Section 4, to realize secure channels between nodes that are not directly connected, but with a connectivity pattern that can be abstracted out as the one considered in this section. First, we recall the properties of SMT.

Definition 4. *A protocol between $\mathcal{S}$ and $\mathcal{R}$ achieves* secure message transmission *if it transmits a message from $\mathcal{S}$ to $\mathcal{R}$ such that the following two conditions are satisfied:*

- CORRECTNESS: *$\mathcal{R}$ learns the message except with probability ε.*
- PRIVACY: *$\mathcal{A}$ does not get any information about the message being transmitted.* $\diamond$

We now describe such protocol. Let $\mathcal{M}$ denote the space of (without loss of generality) q-bit messages. Let ℓ be such that:

1. $\ell \geq q$, and
2. $\ell > c \log \frac{N}{\varepsilon}$, for suitable c (specified later).

We also assume the availability of an error-correcting code tolerating a constant fraction of errors and constant blow-up; for concreteness, say up to $\frac{1}{4}$ of errors can be corrected, and that a q-bit message maps to a $12q$-bit codeword. Let *Enc* and *Dec* be the code's associated functions. The protocol, called PUB-SMT, is shown in Figure 1.

Theorem 1. *Protocol* PUB-SMT*, running on the network described above, is a four-round SMT protocol according to Definition 4, transmitting $O(\max(q, \log \frac{N}{\varepsilon}))$ bits on each of the N channels and $N \cdot O(\max(q, \log \frac{N}{\varepsilon}))$ bits over the public channel.*

Proof

CORRECTNESS: Correctness would not hold if adversary $\mathcal{A}$ is able to corrupt Round 1 messages over any of the N channels and remain undetected. For each channel C_i, this would happen if $\mathcal{A}$ is able to corrupt more than 3ℓ bits. The probability of detecting one of these changes when one bit is revealed in Round 2 is at least $\frac{1}{5}$; thus, the probability that $\mathcal{A}$ remains undetected when 3ℓ bits are revealed is less than $(\frac{4}{5})^{3\ell}$. That's for each individual channel. The probability that $\mathcal{A}$ succeeds on any channel is $N(\frac{4}{5})^{3\ell}$. Setting $N(\frac{4}{5})^{3\ell} < \varepsilon$ yields $\ell > \frac{\log \frac{N}{\varepsilon}}{3 \log \frac{5}{4}} = O(\log \frac{N}{\varepsilon})$.

PRIVACY: Since according to the formulation of the problem, at least one channel (say, channel C_j) remains hidden from the adversary, this channel will always remain in

Protocol PUB-SMT($\mathcal{S}$,$\mathcal{R}$,M,$\mathcal{C}$)

1. $\mathcal{S} \rightarrow \mathcal{R}$: Over each channel $C_i \in \mathcal{C}$, $\mathcal{S}$ sends to $\mathcal{R}$ uniformly chosen random bit string R_i, $|R_i| = 15\ell$. Let R'_i, $1 \leq i \leq N$, be the string received by $\mathcal{R}$ on channel C_i. $\mathcal{R}$ rejects all channels where $|R'_i| \neq 15\ell$.
2. $\mathcal{S} \rightarrow \mathcal{R}$: Let R^*_i denote R_i with 12ℓ randomly chosen positions (for each channel) replaced with "$*$." For each $C_i \in \mathcal{C}$, $\mathcal{S}$ sends R^*_i to $\mathcal{R}$ over *Pub*.
3. $\mathcal{R} \rightarrow \mathcal{S}$: For all channels $C_i \in \mathcal{C}$, if R^*_i and R'_i differ in any of the "opened" bits, $\mathcal{R}$ declares channel C_i as "faulty." I.e., $\mathcal{R}$ sends to $\mathcal{S}$ over *Pub* an N-bit string which identifi es the faulty channels (say, as 0).
   Let $\overline{\mathcal{C}} = \{\overline{C_1}, \overline{C_2}, ..., \overline{C_L}\}$, $L \leq N$, denote the set of remaining, non-faulty channels, and $\overline{R_i}$, $|\overline{R_i}| = 12\ell, 1 \leq i \leq L$, denote the corresponding string of unopened bits; let $\overline{R'_i}$ be the corresponding string in $\mathcal{R}$'s possession.
4. $\mathcal{S} \rightarrow \mathcal{R}$: Let $|M| = q$ (if $|M| < q$, pad M accordingly). For $1 \leq i \leq L$, $\mathcal{S}$ chooses M_i such that $M = M_1 \oplus M_2 \oplus \cdots \oplus M_L$, and sends $S_i = Enc(M_i) \oplus \overline{R_i}$, $1 \leq i \leq L$, over *Pub*.
   For $1 \leq i \leq L$, $\mathcal{R}$ fi rst computes $M'_i = Dec(S_i \oplus \overline{R'_i})$, and then $M' = M'_1 \oplus M'_2 \oplus \cdots \oplus M'_L$ to retrieve the message.

Fig. 1. Protocol for secure message transmission by public discussion

the set of non-faulty channels $\overline{\mathcal{C}}$, and thus any message will be masked by this channel's bits. Hence, for all messages $M_1, M_2 \in \mathcal{M}$ and for all adversaries $\mathcal{A}$, the distribution of $\mathcal{A}$'s view when M_1 is transmitted is identical to the distribution when M_2 is transmitted.

The communication complexity is easily established by inspection. □

The availability of the public channel makes it possible to tolerate a powerful adversary, who is allowed to eavesdrop and/or change the contents of all but one of the N channels. As mentioned at the beginning of the section, our application of SMT by public discussion to almost-everywhere MPC will be to provide secure channels between nodes that are not directly connected in the underlying network, and this section's channels will be instantiated by disjoint paths. Thus, in order to guarantee privacy not only with respect to the adversary, but also with respect to the other honest players, we will be requiring that at least *two*, instead of just one, of the channels (paths) remain untouched (i.e., the corresponding nodes remain uncorrupted) by the adversary. We show how to achieve this in the next section.

4 Almost-Everywhere Secure Multi-party Computation

In this section we first consider graphs with some special properties, which we call *almost-everywhere admissible graphs*, and which will constitute our candidate networks for almost-everywhere MPC. The literature on almost-everywhere agreement [11,30,5] describes several classes of such graphs; however, not all of them are suited to satisfy the privacy requirement of almost-everywhere MPC mentioned above. First, given graphs with degree $d = d(n)$ that allow almost-everywhere agreement — more specifically, almost-everywhere broadcast, we show an explicit transformation to a new graph with degree $O(d)$ satisfying the requirement. We then show a protocol for almost-everywhere MPC on this type of graphs, followed by instantiations of our results on concrete networks.

4.1 Almost-Everywhere Admissible Graphs

First, a more general definition, to succintly express graphs whose sets of privileged nodes have a minimum of uncorrupted paths connectivity as well as a broadcast channel.

Definition 5. *Let $G_n = (V, E)$, $|V| = n$ be a graph, $T \subset V$, $|T| \le t$, $\mathcal{X} : \mathcal{V}^{(t)} \to \mathcal{V}$ a monotically increasing function and $W = V - \mathcal{X}(T)$. We say that G_n is* almost-everywhere (i, t)-admissible *((i, t)-admissible for short) if the following two conditions are satisfied:*

1. *Nodes in W can successfully run almost-everywhere broadcast protocols with polynomial message complexity[2] ; and*
2. *there exists a computable map* Select-Path(G_n, u, v) *outputting a set* PATHS (u, v) *such that*
   (a) *for all $u, v \in V$, $|\text{PATHS}(u, v)| \in O(\text{poly}(n))$;*
   (b) *for all $u, v \in W$, $\text{PATHS}(u, v)$ contains at least i disjoint paths fully contained in W.* ◇

Further, if both procedures in conditions 1 and 2 — the almost-everywhere broadcast protocol and map Select-Path, respectively — are efficiently computable, we call G_n an *efficient* (i, t)-admissible graph.

$(2, t)$-admissible graphs are required by our application as, as mentioned before, two disjoint paths are needed in order to guarantee privacy with respect to intermediate nodes in the paths between nodes, even if those nodes are not corrupted. On the other hand, $(1, t)$-admissible graphs are of particular interest, as there exist constructions for graphs of bounded degree that yield large sets W, while tolerating sets T with the largest sizes, i.e., $|T| = O(n)$ ([30]; see Corollary 4 in Section 4.3). Given that, we now show a transformation to turn $(1, t)$-admissible graphs into $(2, t)$-admissible, while (asymptotically) maintaining the original graphs' desired properties. Recall that we let $X = \max_{T \subset V, |T|=t}\{|\mathcal{X}(T)|\}$.

Lemma 1. *Let $G_n = (V, E)$ and $G'_{2n} = (V', E')$ both be $(1, t)$-admissible graphs according to Definition 5. Then, one can construct a $(2, t)$-admissible graph $G''_{2n} = (V'', E'')$ with subset W'' such that $|W''| \ge 2n - O(X'')$, where $X'' = X + X'$.*

Proof. Graph G''_{2n} is constructed as follows. First, take two copies of G_n, call them G_1 and G_2. Define $V'' = V_1 \bigcup V_2$ and add additional edges between the isomorphic

[2] By "successfully" we mean that for privileged senders (i.e., senders in W) the validity condition is satisfied (see Section 2).

vertices of G_1 and G_2. Note that the resulting graph so far has $2n$ vertices, and $2|E| + |V|$ edges.

Next, order the V'' vertex set in an arbitrary order and add to it all the *edges* from graph G'_{2n}; the resulting edge set is E'', with $|E''| = 2|E|+|V|+|E'|$. For convenience, call G_3 the instance of G'_{2n} applied to G''_{2n}.

We note that we allow *any* (but up to) t nodes to be corrupted in G''_{2n}. We account for every node corrupted in G''_{2n} as two corruptions: one in either (vertex set of) G_1 or G_2, and simultaneously as a corruption in G_3, since G_3 "reuses" the vertex sets of G_1 and G_2.

Now, for a subset of nodes $S_1 \subset V_1$, let $I(S_1)$ be the set of nodes in V_2 isomorphic to the nodes in S_1; define set $I(S_2)$ similarly. Let $T'' = T_1 \bigcup T_2 \bigcup T_3$, and let W_1 (respectively, W_2 and W_3) be the subset of nodes in G_1 (respectively, G_2, G_3) satisfying the premises of the lemma — G_n and G'_{2n} being $(1, t)$-admissible graphs.

Finally, let $W'' = (W_1 - I(T_2) \bigcup W_2 - I(T_1)) \bigcap W_3$. We now show that nodes in W'' can successfully run an almost-everywhere broadcast protocol, and that for all $u, v \in W''$, *two* disjoint paths fully contained in W'' exist connecting them. We have the following cases:

1. $u, v \in V_1 \bigcap W''$: Almost-everywhere broadcast is obtained from G_3, specifically by running the protocol solely on G_3's edges.[3] One path of uncorrupted nodes between nodes u and v is given to us by the premises of the lemma with respect to G_1. The second path is as follows: 1) $u \to u'$, where $u' = I(u)$, 2) $u' \to v'$, where $v' \in I(v)$, and 3) $v' \to v$.
2. $u, v \in V_2 \bigcap W''$: Similar to case 1.
3. $u \in (V_1 \bigcap W'')$ and $v \in (V_2 \bigcap W'')$: Again, almost-everywhere broadcast is given to us by G_3. Let $u' = I(u)$ and $v' = I(v)$. The two paths containing nodes in W'' are as follows:
   (a) $u \to u'$; $u' \rightsquigarrow v$: a path in W'' assumed by the lemma for G_2;
   (b) $u \rightsquigarrow v'$, a path in W'' assumed by the lemma for G_1; $v' \to v$.
4. $u \in (V_2 \bigcap W'')$ and $v \in (V_1 \bigcap W'')$: Similar to case 3.

Let us now estimate the size of the subset W'':

$$\begin{aligned} |W''| &\geq |V''| - |\mathcal{X}(T_1)| - |I(T_2)| - |\mathcal{X}(T_2)| - |I(T_1)| - |\mathcal{X}'(T_3)| \\ &\geq |V''| - 4X - X' \qquad (\text{since } |I(T_i) \leq |\mathcal{X}(T_i)| \leq X, i = 1, 2) \\ &= 2n - O(X''), \end{aligned}$$

where $X'' = X + X'$. □

In the next section we show how to construct X-MPC protocols on $(2, t)$-admissible graphs.

4.2 Almost-Everywhere MPC Protocols

We will be using several building blocks, including protocol PUB-SMT from Section 3, as well as protocols for unconditional VSS (see Section 2.1) and MPC [2], the last two defined on a fully connected network.

[3] This is important, as this property is not preserved under edge addition.

However, first we would like to modify the specification of information-theoretically secure MPC (on a fully connected network and tolerating active adversaries) somewhat, so that it suits our purposes. Typically, the definition postulates an ideal model (equipped with a trusted third party) and compares it to the real model, demanding that in real life the adversary does not gain any advantage compared to what happens in the ideal model [22]. In order to achieve this goal, all known implementations of MPC follow a "commit-and-compute" paradigm. It is convenient for us to recast those results in that paradigm.

Recall that there are n players $P_1, ..., P_n$, each P_i holding a private value x_i, and wishing to jointly compute some function $f(x_1, \cdots, x_n)$. We call the modified protocol C&C-MPC, consisting of two phases:

Commit phase: Players commit to their inputs by acting as dealers in the sharing phase of a $(n, \frac{n}{3})$-VSS protocol — i.e., an unconditional, optimally resilient VSS protocol (e.g., [20,13]). (n executions of the protocol are run in parallel.) At the end of this phase, each player P_i holds a vector of n secret values (shares) $x_i^* = (v_i^1, ...v_i^n)$, one for each VSS invocation.

Computation phase: Players execute the original MPC protocol to compute an "augmented" function f^* defined as the composition of f and n invocations of Rec, the reconstruction function of the VSS protocol:

$$f^*(x_1^*, x_2^*, ..., x_n^*) = f(Rec(v_1^1, v_2^1, ..., v_n^1), Rec(v_1^2, v_2^2, ..., v_n^2), ..., \\ Rec(v_1^n, v_2^n, ..., v_n^n)),$$

where Rec is the reconstruction function of the $(n, \frac{n}{3})$-VSS protocol.

We stress that the Rec protocol is not executed "in the open" as one typically would in an execution of a VSS protocol, but as part of the MPC protocol. Thus, the results of each Rec invocation remain hidden within the MPC computation. Assuming the security of the VSS protocol, it is easy to see that C&C-MPC satisfies the same requirements as the original MPC protocol (correctness, privacy, and independence of inputs).

Having specified this version of MPC, our general approach to almost-everywhere MPC will be to have the players simulate C&C-MPC on the partially connected, admissible network, chosen with a suitable set of parameters, with the following replacement of actions:

1. The sending (and receiving) of messages on the secure channels substituted by invocations to protocol PUB-SMT, and
2. invocations to the public channel (in PUB-SMT) and broadcast (VSS protocol) substituted by invocations to the almost-everywhere broadcast protocol.

We give a more detailed description of the protocol and argue its security below.

Theorem 2. *Let $G_n = (V, E)$ be a $(2, t)$-admissible graph, with T, $\mathcal{X}$ and W as in Definition 5. Let $X = \max_{T \subset V, |T|=t}\{|\mathcal{X}(T)|\}$ and such that $X < \frac{n}{3}$. Then there exists a protocol that achieves X secure multi-party computation against an adaptive, rushing t-adversary.*

Proof sketch. First, we specify the communication structure of the protocol simulation. Each round of protocol C&C-MPC for complete networks is thought of as a "super-round." Each super-round has the same structure, with players taking turns[4] (in, say, lexicographic order) to perform the simulation of sends and receives required in the original round. More specifically, at the onset, each player P_i locally invokes procedure SELECT-PATH(G_n, P_i, P_j), the computable map given by G_n, to obtain set PATHS(P_i, P_j), for every P_j. Whenever P_i is required to send message m to P_j, P_i and P_j run PUB-SMT$(P_i, P_j, m, \text{PATHS}(P_i, P_j))$; invocations to the public channel by P_i (resp., P_j) in PUB-SMT are substituted by invocations to the almost-everywhere broadcast protocol, also given by G_n, with P_i (resp., P_j) acting as the sender. Similarly, invocations by P_i to broadcast in the $(n, \frac{n}{3})$-VSS protocol are replaced by an invocation to the almost-everywhere broadcast protocol with P_i as the sender.

Let $f(x_1, x_2, \cdots, x_n)$ be the function to be computed, where x_i is P_i's private input. Players now simulate the execution of the C&C-MPC protocol: first the commit phase — let $x_1^*, x_2^*, ..., x_n^*$ be the values held by the players at the end of this phase, followed by the computation of the "augmented" function f^*.

First, note that the communication structure of the protocol simulation within the super-round (serialized, one player at a time, in turn one edge at a time) does not introduce any security vulnerabilities, as the original simulated protocols are robust against rushing adversaries, who are allowed to learn the messages sent by the honest players in a round before deciding on the messages for the same round.

We now argue that the conditions of Definition 3, our definition of X secure multi-party computation, are satisfied.

Commitment phase. The premise of the theorem guarantees that $|W| > \frac{2n}{3}$. Thus, it follows from the (simulation of the) sharing phase of the $(n, \frac{n}{3})$-VSS protocol and the properties of almost-everywhere broadcast that for every player $P_i \in V$, there is a value x_i^* uniquely defined by its shares v_i^j, $1 \leq j \leq n$; for players in W in particular, $x_i^* = x_i$, since those players are able to run almost-everywhere broadcast successfully (see Definition 5). We stress that players in $\mathcal{X}(T)$, not only the corrupted ones but also the doomed ones, might provide modified values or not be able to provide any input at all; regardless, they will be unique and well defined per the properties above. This gives the binding property of the commitment phase. The privacy of the input values for players in W follows from the privacy condition of PUB-SMT, which again these players are able to execute successfully, and which guarantees that the views of the adversary — as well as of other honest players, since the graph is $(2, t)$-admissible — under the transmission of any two messages are identical.

Computation phase. Regarding correctness, again since $|W| > \frac{2n}{3}$ and players in W can send each other private messages and simulate broadcast faithfully, they can carry on the reconstruction and the computation on the uniquely defined shared values in the commitment phase, following the protocol for fully connected MPC. Privacy of the computation phase follows from a hybrid argument and reduction to the privacy of the message transmission scheme. In a fully connected network, the condition of

[4] This for simplicity, and to avoid a more detailed analysis of possible interference. Techniques from [5] could in principle be applied in order to reduce the total number of rounds.

indistinguishable views for the adversary for all $\overrightarrow{x^*_{V-T}}$, $\overrightarrow{y^*_{V-T}}$, $\overrightarrow{z^*_T}$ such that the output of the function is the same, i.e., $f(\overrightarrow{x^*_{V-T}}, \overrightarrow{z^*_T}) = f(\overrightarrow{y^*_{V-T}}, \overrightarrow{z^*_T})$, is known to hold for an information-theoretically secure MPC protocol as long as the sets of corrupted players are the same [2]. Thus, if the adversary would be able to distinguish the two views with non-negligible advantage in the simulated execution, then there would be a particular super-round — in turn, player turn; in turn, message transmission — where the adversary can distinguish the two runs on G_n, but does not distinguish them in the fully connected network. This would contradict the security of the message transmission protocol between two privileged players. □

4.3 Almost-Everywhere MPC on Classes of Networks

In this section we enumerate several classes of networks where almost-everywhere MPC is possible, as a corollary of admissible graphs given in the almost-everywhere agreement literature. The first three corollaries follow from results in [11], and the last one from [30].

Corollary 1. *For all $r \geq 5$, almost all r-regular graphs (i.e., all but a vanishingly small fraction of such graphs) admit $O(t)$-MPC, where $|T| = t \leq n^{1-c}$, for some constant $c = c(r)$, where $c(r) \to 0$ as $r \to n$.*

The next corollaries apply to explicit graphs for which the number of doomed players is small.

Corollary 2. *For every $\epsilon > 0$ there exists a network $G_n = (V, E)$ of degree $O(n^\epsilon)$ and $t = O(n)$ on which $O(t)$-MPC is possible.*

The corollary follows from a recursive construction of networks of unbounded degree in [11] that yields $(2, O(n))$-admissible graphs with $X = O(t)$.

Corollary 3. *There exists a constant-degree network with $t = O(\frac{n}{\log n})$ on which $O(t)$-MPC is possible.*

This network is constructed by taking a butterfly network, which constitutes a $(2, O(\frac{n}{\log n}))$-admissible graph, with $X = O(t \log t)$, and superimposing a 5-regular graph; this yields a regular graph of degree 9, on which a *compression* procedure can be run to “sharpen” the X term to $O(t)$ [11].

Finally, Upfal [30] shows how to explicitly construct constant-degree expander graphs that yield $(1, O(n))$-admissible graphs — i.e., tolerating large (*linear*) number of corruptions (compare to the other constant-degree networks above) — while avoiding the blow-up in the number of doomed players. Applying the construction given in Lemma 1, we obtain:

Corollary 4. *There exist constant-degree networks with $t = O(n)$ on which $O(t)$-MPC is possible.*

The protocol achieving it, however, is not efficient (i.e., polynomial-time), as the resulting admissible graph is not efficient; specifically, the almost-everywhere broadcast component has polynomial message complexity but requires exponential computation.

5 Summary and Future Work

In this paper we introduced the notion of almost-everywhere secure multi-party computation for partially connected networks, and showed how to achieve meaningful security guarantees whenever possible. We proposed a definition for X-MPC, and a protocol satisfying it. We also gave concrete examples for specific networks, building on work from almost-everywhere agreement.

Regarding our definitional approach, which follows the one in [23], it is well known that simulation-based definitions of security are stronger than and preferable to indistin guishability-based ones. However, in the setting of almost-everywhere secure computation, the simulation-based approach encounters the following problem: it seems challenging how to define, in a meaningful and network-independent way, the simulation and the adversarial view of the state of the doomed players, or indeed how to even deal with this dynamically growing set; even though these nodes are not part of the nodes for which we guarantee privacy and a correct output, it is not clear what view of these nodes the adversary gets. Indeed, for some of the doomed nodes the adversary could learn all the information and be able to change their inputs, while for others the adversary would only get partial control. We leave the refinement of and alternatives to our almost-everywhere MPC definition as a subject for future research. We stress though that in many situations, the security guarantees given by our approach are sufficient, especially if running information-theoretically secure protocols, such as the one in [2].

Regarding our new model for SMT by public discussion, it would be interesting to reduce the communication, in particular on the public channel (say, to sublinear in N), and provide some measure of optimality.

Finally, providing a polynomial-time protocol for almost-everywhere agreement — and thus for almost-everywhere MPC — on networks of bounded degree tolerating a linear number of corruptions remains an interesting open problem.

Acknowledgments

We thank the anonymous reviewers for *Eurocrypt 2008* for their useful comments.

The problem statement of almost-everywhere secure computation, as well as the overall approach are joint work with Shailesh Vaya, as reflected in [31,17,18][5]. However, the simulation-based approach to security that was originally considered in the three-author draft, and further developed in [32], we (the current authors) eventually found unsatisfactory (see Sections 1, paragraph on related work, and 5 for a more technical discussion). Hence, we withdrew our names from that manuscript, and proceeded to work out a different model and corresponding proof of security, as well as to completely change and overhaul the protocols — this is the work presented here. As a courtesy, Vaya was offered co-authorship on the current paper, provided however that the problematic approach be abandoned and not published as a separate work; he chose not to accept our offer.

[5] In [31], Shailesh Vaya's thesis acknowledgement reads: *"Here, let me also thank Juan Garay who along with Rafail was an equal participant in developing the central ideas in this thesis."*

References

1. Agarwal, S., Cramer, R., de Haan, R.: Asymptotically optimal two-round perfectly secure message transmission. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proc. 20th Annual ACM Symposium of the Theory of Computation, pp. 1–10 (May 1988)
3. Bennett, C.H., Brassard, G., Crèpeau, C., Maurer, U.: Generalized privacy amplification. IEEE Transactions on Information Theory 41(6), 1015–1923 (1995)
4. Bennett, C.H., Brassard, G., Robert, J.M.: Privacy amplification by public discussion. Siam Journal of Computing 17(2) (April 1988)
5. Berman, P., Garay, J.: Fast consensus in networks of bounded degree. WDAG 1990 2(7), 62–73 (1993); Preliminary version in WDAG 1990
6. Chaum, D., Crepeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: Proceedings 20th Annual Symposium on Theory of Computing, STOC, Association for Computing Machinery (May 1988)
7. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and simultaneous broadcast. In: Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, pp. 383–395 (1985)
8. Desmedt, Y., Wang, Y.: Perfectly secure message transmission revisited. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 502–517. Springer, Heidelberg (2002)
9. Dolev, D.: The byzantine generals strike again. Journal of Algorithms 1(3), 14–30 (1982)
10. Dolev, D., Dwork, C., Waarts, O., Young, M.: Perfectly secure message transmission. Journal of ACM 1(40), 17–47 (1993)
11. Dwork, C., Peleg, D., Pippinger, N., Upfal, E.: Fault tolerance in networks of bounded degree. In: Proc. 18th Annual Symposium on the Theory of Computing, pp. 370–379 (1986)
12. Fitzi, M., Franklin, M., Garay, J., Vardhan, S.H.: Towards optimal and efficient perfectly secure message transmission. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (2007)
13. Fitzi, M., Garay, J., Gollakota, S., Rangan, C.P., Srinathan, K.: Round-optimal and efficient verifiable secret sharing. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 329–342. Springer, Heidelberg (2006)
14. Fitzi, M., Hirt, M., Maurer, U.: Trading correctness for privacy in unconditional mpc. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, Springer, Heidelberg (1998)
15. Franklin, M.K., Wright, R.N.: Secure communications in minimal connectivity models. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 346–360. Springer, Heidelberg (1998)
16. Garay, J., Moses, Y.: Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. SIAM J. Comput. 27(1), 247–290 (1998); Preliminary version in STOC 1992
17. Garay, J., Ostrovsky, R., Vaya, S.: Almost-eveywhere secure computation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (2007)
18. Garay, J., Ostrovsky, R., Vaya, S.: Almost-eveywhere secure computation. In: Presentation at the 2007 Workshop on Cryptographic Protocols WCP 2007, Bertinoro, Italy (March 2007)
19. Garay, J., Perry, K.: A continuum of failure models for distributed computing. In: Segall, A., Zaks, S. (eds.) WDAG 1992. LNCS, vol. 647, pp. 153–165. Springer, Heidelberg (1992)
20. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: On 2-round secure multiparty computation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, Springer, Heidelberg (2002)
21. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for orotocols with honest majority. In: Proc. 19th Annual ACM Symposium on Theory of Computation, pp. 218–229 (May 1987)

22. Goldreich, O.: Secure multi-party computation, final (incomplete) draft, version 1.4 (2002)
23. Kilian, J., Kushilevitz, E., Micali, S., Ostrovsky, R.: Reducibility and completeness in private computations. SIAM Journal on Computing 29 (1999)
24. Kurosawa, K., Suzuki, K.: Truly efficient 2-round perfectly secure message transmission scheme. These proceedings
25. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Transactions on Programming Languages and Systems, 382–401 (July 1982)
26. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM, JACM 27(2) (April 1980)
27. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proc. 21st ACM Symposium on the Theory of Computing, pp. 73–85 (1989)
28. Sayeed, H., Abu-Amara, H.: Efficient perfectly secure message transmission in synchronous networks. Information and Computation 1(126), 53–61 (1996)
29. Srinathan, K., Narayanan, A., Pandu Rangan, C.: Optimal perfectly secure message transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004)
30. Upfal, E.: Tolerating linear number of faults in networks of bounded degree. In: Proc. 11th ACM Symposiyum on Principles of Distributed Computing, pp. 83–89 (1992)
31. Vaya, S.: Almost-everywhere secure computation. Ph.D. Thesis, University of California at Los Angeles, California (December 2006)
32. Vaya, S.: Secure computation on incomplete networks. In: Cryptology ePrint archive, Report 2007/346 (September 2007)
33. Yao, A.: Protocols for secure computation. In: Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (1982)

Truly Efficient 2-Round Perfectly Secure Message Transmission Scheme

Kaoru Kurosawa[1] and Kazuhiro Suzuki[2]

[1] Department of Computer and Information Sciences, Ibaraki University
kurosawa@mx.ibaraki.ac.jp
[2] Venture Business Laboratory, Ibaraki University, Hitachi, Ibaraki 316-8511, Japan
tutetuti@dream.com

Abstract. In the model of perfectly secure message transmission schemes (PSMTs), there are n channels between a sender and a receiver. An infinitely powerful adversary $\mathbf{A}$ may corrupt (observe and forge) the messages sent through t out of n channels. The sender wishes to send a secret s to the receiver perfectly privately and perfectly reliably without sharing any key with the receiver.

In this paper, we show the first 2-round PSMT for $n = 2t + 1$ such that not only the transmission rate is $O(n)$ but also the computational costs of the sender and the receiver are both polynomial in n. This means that we solve the open problem raised by Agarwal, Cramer and de Haan at CRYPTO 2006.

Keywords: Perfectly secure message transmission, information theoretic security, efficiency.

1 Introduction

In the model of (r-round, n-channel) message transmission schemes [2], there are n channels between a sender and a receiver. An infinitely powerful adversary $\mathbf{A}$ may corrupt (observe and forge) the messages sent through t out of n channels. The sender wishes to send a secret s to the receiver in r-rounds without sharing any key with the receiver.

We say that a message transmission scheme is perfectly secure if it satisfies perfect privacy and perfect reliability. The perfect privacy means that the adversary $\mathbf{A}$ learns no information on s, and the perfect reliability means that the receiver can output $\hat{s} = s$ correctly.

For $r = 1$, Dolev et al. showed that there exists a 1-round perfectly secure message transmission scheme (PSMT) if and only if $n \geq 3t + 1$ [2]. They also showed an efficient 1-round PSMT [2].

For $r \geq 2$, it is known that there exists a 2-round PSMT if and only if $n \geq 2t + 1$ [2]. However, it is very difficult to construct an efficient scheme for $n = 2t + 1$. Dolev et al. [2] showed a 3-round PSMT such that the transmission rate is $O(n^5)$, where the transmission rate is defined as

$$\frac{\text{the total number of bits transmitted}}{\text{the size of the secrets}}.$$

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 324–340, 2008.

Sayeed et al. [7] showed a 2-round PSMT such that the transmission rate is $O(n^3)$.

Recently, Srinathan et al. showed that n is a lower bound on the transmission rate of 2-round PSMT [8]. Then Agarwal, Cramer and de Haan [1] showed a 2-round PSMT such that the transmission rate is $O(n)$ at CRYPTO 2006 based on the work of Srinathan et al. [8][1]. However, the communication complexity is exponential because the sender must broadcast consistency check vectors of size[2]

$$w = \binom{n-1}{t+1} = \binom{2t}{t+1}.$$

In other words, Agarwal et al. [1] achieved the transmission rate of $O(n)$ by sending exponentially many secrets. Therefore, the computational costs of the sender and the receiver are both exponential. Indeed, the authors wrote [1, Sec.6] that:

"We do not know whether a similar protocol can exist where sender and receiver restricted to polynomial time (in terms of the number of channels n) only".

In this paper, we solve this open problem. That is, we show the first 2-round PSMT for $n = 2t + 1$ such that not only the transmission rate is $O(n)$ but also the computational costs of the sender and the receiver are both polynomial in n.

Table 1. 2-Round PSMT for $n = 2t + 1$

	Trans. rate	com. complexity	Receiver	Sender
Agarwal et al. [1]	$O(n)$	exponential	exponential	exponential
This paper	$O(n)$	$O(n^3)$	poly	poly

The main novelty of our approach is to introduce a notion of *pseudo-basis* to the coding theory. Let $\mathcal{C}$ be a linear code of length n over a finite field F with the minimum Hamming distance $d = t+1$. Consider a message transmission scheme such that the sender chooses a codeword $X_i = (x_{i1}, \cdots, x_{in})$ of $\mathcal{C}$ randomly and sends x_{ij} through channel j for $j = 1, \cdots, n$. Note that the receiver can detect t errors, but cannot correct them because $d = t + 1$.

If the sender sends many codewords, however, then we can do something better. Suppose that the sender sent X_i as shown above, and the receiver received $Y_i = X_i + E_i$ for $i = 1, \cdots, m$, where E_i is an error vector caused by the adversary. We now observe that the dimension of the space $\mathcal{E}$ spanned by the error vectors $E_1, \cdots, E_m$ is at most t because the adversary corrupts at most t channels. Suppose that $\{E_{i_1}, \cdots, E_{i_k}\}$ is such a basis, where $k \leq t$. For the same indices,

[1] Srinathan et al. claimed that they constructed a 2-round PSMT such that the transmission rate is $O(n)$ in [8]. However, Agarwal et al. pointed out that it has a flaw in [1].

[2] Indeed, in [1, page 407], it is written that "at most $O(w)$ indices and field elements are broadcast ....", where w is defined in [1, page 403] as shown above.

we say that $\mathcal{B} = \{Y_{i_1}, \cdots, Y_{i_k}\}$ is a *pseudo-basis* of $\mathcal{Y} = \{Y_1, \cdots, Y_m\}$. We then show that a receiver can find a pseudo-basis $\mathcal{B}$ of $\mathcal{Y}$ in polynomial time.

By using this algorithm, we first show a 3-round PSMT for $n = 2t + 1$ such that the transmission rate is $O(n)$ and the computational cost of the sender and the receiver are both polynomial in n. (See Fig.2.5.) Then combining the technique of [8,1], we show a 2-round PSMT such that not only the transmission rate is $O(n)$ but also the computational cost of the sender and the receiver are both polynomial in n.

(Remark) Recently, Fitzi et al. showed an efficient 2-round PSMT for $n \geq (2+\epsilon)t$ for any constant $\epsilon > 0$ [4], but not for $n = 2t + 1$.

2 Main Idea

Suppose that there are n channels between the sender and the receiver, and an adversary may corrupt t out of n channels. We use F to denote $GF(p)$, where p is a prime such that $p > n$.[3] Let $\mathcal{C}$ be a linear code of length n such that a codeword is $X = (f(1), \cdots, f(n))$, where $f(x)$ is a polynomial over F with $\deg f(x) \leq t$.

2.1 Difference from Random *t* Errors

Consider a message transmission scheme such that the sender chooses a codeword $X = (f(1), \cdots, f(n))$ of $\mathcal{C}$ randomly, and sends $f(i)$ through channel i for $i = 1, \cdots, n$. Then the adversary learns no information on $f(0)$ even if she observes t channels because $\deg f(x) \leq t$. Thus perfect privacy is satisfied.

If $n = 3t+1$, then the minimum Hamming distance of $\mathcal{C}$ is $d = n - t = 2t+1$. Hence the receiver can correct t errors caused by the adversary. Thus perfect reliability is also satisfied. Therefore we can obtain a 1-round PSMT easily.

If $n = 2t+1$, however, the minimum Hamming distance of $\mathcal{C}$ is $d = n-t = t+1$. Hence the receiver can only detect t errors, but cannot correct them. This is the main reason why the construction of PSMT for $n = 2t + 1$ is difficult.

What is a difference between usual error correction and PSMTs ? If the sender sends a single codeword $X \in \mathcal{C}$ only, then the adversary causes t errors randomly. Hence there is no difference. If the sender sends many codewords $X_1, \cdots, X_m \in \mathcal{C}$, however, the errors are not totally random. This is because the errors always occur at the same t (or less) places !

To see this more precisely, suppose that the receiver received

$$Y_i = X_i + E_i, \tag{1}$$

where $E_i = (e_{i1}, \cdots, e_{in})$ is an error vector caused by the adversary. Define

$$support(E_i) = \{j \mid e_{ij} \neq 0\}.$$

[3] We adopt $GF(p)$ only to make the presentation simpler, where the elements are denoted by $0, 1, 2, \cdots$. But in general, our results hold for any finite field F whose size is larger than n.

Then there exist some t-subset $\{j_1, \cdots, j_t\}$ of n channels such that each error vector E_i satisfies

$$support(E_i) \subseteq \{j_1, \cdots, j_t\}, \tag{2}$$

where $\{j_1, \cdots, j_t\}$ is the set of channels that the adversary forged.

This means that the space $\mathcal{E}$ spanned by $E_1, \cdots, E_m$ has dimension at most t. We will exploit this fact extensively.

2.2 Pseudo-basis and Pseudo-dimension

Let $\mathcal{V}$ denote the n-dimensional vector space over F . For two vectors $Y, E \in \mathcal{V}$, we write

$$Y = E \bmod \mathcal{C}$$

if $Y - E \in \mathcal{C}$.

For $i = 1, \cdots, m$, suppose that the receiver received Y_i such that

$$Y_i = X_i + E_i,$$

where $X_i \in \mathcal{C}$ is a codeword that the sender sent and E_i is the error vector caused by the adversary. From now on, (Y_i, X_i, E_i) has this meaning. Then we have that

$$Y_i = E_i \bmod \mathcal{C} \tag{3}$$

for each i. Let $\mathcal{E}$ be a subspace spanned by $E_1, \cdots, E_m$.

We first define a notion of *pseudo-span*.

Definition 1. *We say that* $\{Y_{j1}, \cdots, Y_{jk}\} \subset \mathcal{Y}$ *pseudo-spans* $\mathcal{Y}$ *if each* $Y_i \in \mathcal{Y}$ *can be written as*

$$Y_i = a_1 Y_{j1} + \cdots + a_k Y_{jk} \bmod \mathcal{C}$$

for some $a_i \in \mathsf{F}$.

We next define a *pseudo-basis* and the *pseudo-dimension* of $\mathcal{Y}$.

Definition 2

- *Let k be the dimension of $\mathcal{E}$. We then say that $\mathcal{Y}$ has the pseudo-dimension k.*
- *Let $\{E_{j1}, \cdots, E_{jk}\}$ be a basis of $\mathcal{E}$. For the same indices, we say that $\{Y_{j1}, \cdots, Y_{jk}\}$ is a pseudo-basis of $\mathcal{Y}$.*

The following theorem is clear since the adversary forges at most t channels.

Theorem 1. *The pseudo-dimension of* $\mathcal{Y}$ *is at most* t.

Suppose that $\{Y_{j1}, \cdots, Y_{jk}\}$ is a pseudo-basis of $\mathcal{Y}$. Define

$$\mathsf{FORGED} = \bigcup_{i=1}^{k} support(E_{ji}). \tag{4}$$

It is then clear that FORGED is the set of all channels that the adversary forged. Therefore, the following theorem holds.

Theorem 2. *For each j,*

$$support(E_j) \subseteq \mathsf{FORGED}.$$

We finally prove the following theorem.

Theorem 3. *$\mathcal{B} = \{Y_{j1}, \cdots, Y_{jk}\}$ is a pseudo-basis of $\mathcal{Y}$ if and only if $\mathcal{B}$ is a minimal subset of $\mathcal{Y}$ which pseudo-spans $\mathcal{Y}$.*

(Proof) (I) Suppose that $\mathcal{B}$ is a minimal subset of $\mathcal{Y}$ which pseudo-spans $\mathcal{Y}$. Then each $Y_i \in \mathcal{Y}$ can be written as

$$Y_i = a_1 Y_{j1} + \cdots + a_k Y_{jk} \bmod \mathcal{C}$$

for some $a_i \in \mathsf{F}$. From eq.(3), we obtain that

$$E_i = a_1 E_{j1} + \cdots + a_k E_{jk} \bmod \mathcal{C}.$$

Hence

$$E_i - a_1 E_{j1} - \cdots - a_k E_{jk} \in \mathcal{C}.$$

The Hamming weight of the left hand side is at most t while the minimum Hamming weight of $\mathcal{C}$ is $t+1$. Therefore, $E_i - a_1 E_{j1} - \cdots - a_k E_{jk}$ is a zero-vector. Hence we obtain that

$$E_i = a_1 E_{j1} + \cdots + a_k E_{jk}.$$

This means that $\{E_{j1}, \cdots, E_{jk}\}$ spans $\mathcal{E}$. Further the minimality of $\mathcal{B}$ implies that $\{E_{j1}, \cdots, E_{jk}\}$ is a basis of $\mathcal{E}$. Therefore, from Def.2, $\mathcal{B} = \{Y_{j1}, \cdots, Y_{jk}\}$ is is a pseudo-basis of $\mathcal{Y}$.

(II) Suppose that $\mathcal{B} = \{Y_{j1}, \cdots, Y_{jk}\}$ is a pseudo-basis of $\mathcal{Y}$. Then $\{E_{j1}, \cdots, E_{jk}\}$ is a basis of $\mathcal{E}$. Therefore each E_i can be written as

$$E_i = a_1 E_{j1} + \cdots + a_k E_{jk}$$

for some $a_i \in \mathsf{F}$. This means that each Y_i is written as

$$Y_i = a_1 Y_{j1} + \cdots + a_k Y_{jk} \bmod \mathcal{C}$$

from eq.(3). Hence $\mathcal{B}$ pseudo-spans $\mathcal{Y}$. If $\mathcal{B}$ is not minimal, then we can show that a smaller subset of $\{E_{j1}, \cdots, E_{jk}\}$ is a basis of $\mathcal{E}$. This is a contradiction. Therefore, $\mathcal{B}$ is a minimal subset of $\mathcal{Y}$ which pseudo-spans $\mathcal{Y}$. Q.E.D.

2.3 How to Find Pseudo-basis

In this subsection, we show a polynomial time algorithm which finds the pseudo-dimension k and a pseudo-basis $\mathcal{B} = \{B_1, \cdots, B_k\}$ of $\mathcal{Y} = \{Y_1, \cdots, Y_m\}$. We begin with a definition of *linearly pseudo-express*.

Definition 3. *We say that Y is linearly pseudo-expressed by $\{B_1, \cdots, B_k\}$ if*

$$Y = a_1 B_1 + \cdots + a_k B_k \bmod \mathcal{C}$$

for some $a_1 \cdots, a_k \in \mathsf{F}$.

We first show in Fig.2.3 a polynomial time algorithm which checks if Y is linearly pseudo-expressed by $\{B_1, \cdots, B_k\}$. For a parameter $\alpha = (a_1 \cdots, a_k)$, define $X(\alpha)$ as

$$\begin{aligned} X(\alpha) &= Y - (a_1 B_1 + \cdots + a_k B_k) \\ &= (x_1(\alpha), \cdots, x_n(\alpha)). \end{aligned} \tag{5}$$

From the definition, Y is linearly pseudo-expressed by $\{B_1, \cdots, B_k\}$ if and only if there exists some α such that $X(\alpha) \in \mathcal{C}$. It is clear that $x_j(\alpha)$ is a linear expression of $(a_1 \cdots, a_k)$ from eq.(5). In Fig.2.3, it is also easy to see that each coefficient of $f_\alpha(x)$ is a linear expression of $(a_1 \cdots, a_k)$. Hence $f_\alpha(j) = x_j(\alpha)$ is a linear equation on $(a_1 \cdots, a_k)$ at step 3.

It is now clear that the algorithm of Fig.2.3 outputs YES if and only if $X(\alpha) \in \mathcal{C}$ for some α. Hence it outputs YES if and only if Y is linearly pseudo-expressed by $\{B_1, \cdots, B_k\}$.

Input: Y and $\mathcal{B} = \{B_1, \cdots, B_k\}$.

1. Construct $X(\alpha) = (x_1(\alpha), \cdots, x_n(\alpha))$ of eq.(5).
2. Construct a polynomial $f_\alpha(x)$ with $\deg f_\alpha(x) \le t$ such that
$$f_\alpha(i) = x_i(\alpha)$$
for $i = 1, \cdots, t+1$ by using Lagrange formula.
3. Output YES if the following set of linear equations has a solution α.
$$\begin{aligned} f_\alpha(t+2) &= x_{t+2}(\alpha), \\ &\vdots \\ f_\alpha(n) &= x_n(\alpha). \end{aligned}$$
Otherwise output NO.

Fig. 1. How to Check if Y is linearly pseudo-expressed by $\mathcal{B}$

We finally show in Fig.2 a polynomial time algorithm which finds the pseudo-dimension k and a pseudo-basis $\mathcal{B} = \{B_1, \cdots, B_k\}$ of $\mathcal{Y} = \{Y_1, \cdots, Y_m\}$. The correctness of the algorithm is guaranteed by Theorem 3.

2.4 Broadcast

We say that a sender (receiver) broadcasts x if it she sends x over all n channels. Since the adversary corrupts at most t out of $n = 2t + 1$ channels, the receiver (sender) receives x correctly from at least $t + 1$ channels. Therefore, the receiver (sender) can accept x correctly by taking the majority vote.

Input: $\mathcal{Y} = \{Y_1, \cdots, Y_m\}$.

1. Let $i = 1$ and $\mathcal{B} = \emptyset$.
2. While $i \leq m$ and $|\mathcal{B}| < t$, do:
(a) Check if Y_i is linearly pseudo-expressed by $\mathcal{B}$ by using Fig.2.3.
If NO, then add Y_i to $\mathcal{B}$.
(b) Let $i \leftarrow i + 1$.
3. Output $\mathcal{B}$ as a pseudo-basis and $k = |\mathcal{B}|$ as the pseudo-dimension.

Fig. 2. How to Find a Pseudo-Basis $\mathcal{B}$ of $\mathcal{Y}$

2.5 How to Apply to 3-Round PSMT

We now present an efficient 3-round PSMT for $n = 2t + 1$ in Fig.2.5.

The sender wishes to send $\ell = nt$ secrets $s_1, \cdots, s_\ell \in \mathsf{F}$ to the receiver.

1. The sender sends a random codeword $X_i = (f_i(1), \cdots, f_i(n))$, and the receiver receives $Y_i = X_i + E_i$ for $i = 1, \cdots, \ell + t$, where $\deg f_i(x) \leq t$ and E_i is the error vector caused by the adversary.
2. The receiver finds a pseudo-basis $\mathcal{B} = \{Y_{j1}, \cdots, Y_{jk}\}$, where $k \leq t$, by using the algorithm of Fig.2.
He then broadcasts $\mathcal{B}$ and $\Lambda_{\mathcal{B}} = \{j_1, \cdots, j_k\}$.
3. The sender constructs FORGED of eq.(4) from $\{E_j = Y_j - X_j \mid j \in \Lambda_{\mathcal{B}}\}$, encrypts $s_1, \cdots, s_\ell$ by using $\{f_i(0) \mid i \notin \Lambda_{\mathcal{B}}\}$ as the key of one-time pad, and then broadcasts FORGED and the ciphertexts.
4. The receiver reconstructs $f_i(x)$ by ignoring all channels of FORGED, and applying Lagrange formula to the remaining elements of Y_i.
He then decrypts the ciphertexts by using $\{f_i(0) \mid i \notin \Lambda_{\mathcal{B}}\}$.

Fig. 3. Our 3-round PSMT for $n = 2t + 1$

Further by combining the technique of [8,1], we can construct a 2-round PSMT such that not only the transmission rate is $O(n)$, but also the computational cost of the sender and the receiver are both polynomial in n. The details will be given in the following sections.

3 Details of Our 3-Round PSMT

In this section, we describe the details of our 3-round PSMT for $n = 2t + 1$ which was outlined in Sec.2.5, and prove its security. We also show that the transmission rate is $O(n)$ and the computational cost of the sender and the receiver are both polynomial in n.

Remember that FORGED is the set of all channels which the adversary forged, and "broadcast" is defined in Sec.2.4.

3.1 3-Round Protocol for $n = 2t + 1$

The sender wishes to send $\ell = nt$ secrets $s_1, \cdots, s_\ell \in \mathsf{F}$ to the receiver.

Step 1. The sender does the following for $i = 1, 2, \cdots, t + \ell$.

1. She chooses a polynomial $f_i(x)$ over F such that $\deg f_i(x) \leq t$ randomly. Let $X_i = (f_i(1), \cdots, f_i(n))$.
2. She send $f_i(j)$ through channel j for $j = 1, \cdots, n$. The receiver then receives $Y_i = X_i + E_i$, where E_i is the error vector caused by the adversary.

Step 2. The receiver does the following.

1. Find the pseudo-dimension k and a pseudo-basis $\mathcal{B} = \{Y_{j1}, \cdots, Y_{jk}\}$ of $\{Y_1, \cdots, Y_{t+\ell}\}$ by using the algorithm of Fig.2.
2. Broadcast k, $\mathcal{B}$ and $\Lambda_{\mathcal{B}} = \{j_1, \cdots, j_k\}$. where $\Lambda_{\mathcal{B}}$ is the set of indices of $\mathcal{B}$.

Step 3. The sender does the following.

1. Construct FORGED of eq.(4) from $\{E_j = Y_j - X_j \mid j \in \Lambda_{\mathcal{B}}\}$.
2. Compute $c_1 = s_1 + f_{i_1}(0), \cdots, c_\ell = s_\ell + f_{i_\ell}(0)$ for $i_1, \cdots, i_\ell \notin \Lambda_{\mathcal{B}}$.
3. Broadcast FORGED and $(c_1, \cdots, c_\ell)$.

Step 4. The receiver does the following. Let $Y_i = (y_{i1}, \cdots, y_{in})$.

1. For each $i \notin \Lambda_{\mathcal{B}}$, find a polynomial $f'_i(x)$ with $\deg f'_i(x) \leq t$ such that
$$f'_i(j) = y_{i,j}$$
for all $j \notin \mathsf{FORGED}$.
2. Compute $s'_1 = c_1 - f'_{i_1}(0), \cdots, s'_\ell = c_\ell - f'_{i_\ell}(0)$ for $i_1, \cdots, i_\ell \notin \Lambda_{\mathcal{B}}$.
3. Output $(s'_1, \cdots, s'_\ell)$.

3.2 Security

We first prove the perfect privacy. Consider $f_i(x)$ such that $i \notin \Lambda_{\mathcal{B}}$. For such i, Y_i is not broadcast at step 2-2. Hence the adversary observes at most t elements of $(f_i(1), \cdots, f_i(n))$. This means that she has no information on $f_i(0)$ because $\deg f_i(x) \leq t$. Therefore since $\{f_i(0) \mid i \notin \Lambda_{\mathcal{B}}\}$ is used as the key of one-time-pad, the adversary learns no information on $s_1, \cdots, s_\ell$.

We next prove the perfect reliability. We first show that there exist ℓ indices $i_1, i_2, \cdots, i_\ell$ such that
$$\{i_1, i_2, \cdots, i_\ell\} \subseteq \{1, 2, \cdots, t + \ell\} \setminus \Lambda_{\mathcal{B}}.$$
This is because
$$t + \ell - |\Lambda_{\mathcal{B}}| \geq t + \ell - t = \ell.$$
from Theorem 1. We next show that $f'_i(x) = f_i(x)$ for each $i \notin \Lambda_{\mathcal{B}}$ at Step 4. This is because
$$f'_i(j) = y_{i,j} = x_{i,j} = f_i(j)$$
for all $j \notin \mathsf{FORGED}$, and
$$n - |\mathsf{FORGED}| \geq 2t + 1 - t \geq t + 1.$$
Also note that $\deg f_i(x) \leq t$ and $\deg f'_i(x) \leq t$. Therefore $s'_i = s_i$ for $i = 1, \cdots, \ell$.

3.3 Efficiency

Let $|\mathsf{F}|$ denote the bit length of the field elements. Let $\mathsf{COM(i)}$ denote the communication complexity of Step i for $i = 1, 2, 3$. Then

$$\begin{aligned}\mathsf{COM}(1) &= O(n(t+\ell))|\mathsf{F}|) = O(n\ell|\mathsf{F}|),\\ \mathsf{COM}(2) &= O(n^2 t|\mathsf{F}|) = O(n\ell|\mathsf{F}|),\\ \mathsf{COM}(3) &= O(n\ell|\mathsf{F}| + tn\log_2 n) = O(n\ell|\mathsf{F}|)\end{aligned}$$

since $\ell = nt$. Hence the total communication complexity is $O(n\ell|\mathsf{F}|) = O(n^3|\mathsf{F}|)$. Further the sender sends ℓ secrets $s_1, \cdots, s_\ell \in \mathsf{F}$. Therefore, the transmission rate is $O(n)$ because

$$\frac{n\ell|\mathsf{F}|}{\ell|\mathsf{F}|} = n.$$

It is easy to see that the computational costs of the sender and the receiver are both polynomial in n.

4 Our Basic 2-Round PSMT

In this section, we show our basic 2-round PSMT for $n = 2t + 1$ such that the transmission rate is $O(n^2 t)$ and the computational costs of the sender and the receiver are both polynomial in n.

For two vectors $U = (u_1, \cdots, u_n)$ and $Y = (y_1, \cdots, y_n)$, define

$$\begin{aligned}d_u(U, Y) &= \{u_j \mid u_j \neq y_j\}\\ d_I(U, Y) &= \{j \mid u_j \neq y_j\}.\end{aligned}$$

Remember that $\mathcal{C}$ is the set of all $(f(1), \cdots, f(n))$ such that $\deg f(x) \leq t$.

4.1 Randomness Extractor

Suppose that the adversary has no information on ℓ out of m random elements $r_1, \cdots, r_m \in \mathsf{F}$. In this case, let $R(x)$ be a polynomial with $\deg R(x) \leq m - 1$ such that $R(i) = r_i$ for $i = 1, \cdots, m$. Then it is well known [1, Sec.2.4] that the adversary has no information on

$$z_1 = R(m+1), \cdots, z_\ell = R(m+\ell).$$

4.2 Basic 2-Round Protocol

The sender wishes to send a secret $s \in \mathsf{F}$ to the receiver.

Step 1. The receiver does the following for $i = 1, 2, \ldots, n$.

1. He chooses a random polynomial $f_i(x)$ such that $\deg f_i(x) \leq t$.

2. He sends

$$X_i = (f_i(1), \cdots, f_i(n))$$

through channel i, and the sender receives

$$U_i = (u_{i1}, \ldots, u_{in}).$$

3. Through each channel j, he sends $f_i(j)$ and the sender receives

$$y_{ij} = f_i(j) + e_{ij},$$

where e_{ij} is the error caused by the adversary. Let

$$Y_i = (y_{i1}, \cdots, y_{in}),\ E_i = (e_{i1}, \cdots, e_{in}).$$

Step 2. The sender does the following.

1. For $i = 1, \cdots, n$,
   (a) If $u_{ii} \neq y_{ii}$ or $|d_u(U_i, Y_i)| \geq t + 1$ or $U_i \notin \mathcal{C}$, then broadcast "ignore channel i".[4]
   This channel will be ignored from now on because it is forged clearly.
   (b) Else define r_i as
$$r_i = u_{ii} = y_{ii}. \tag{6}$$
2. Find a polynomial $R(x)$ with $\deg R(x) \leq n - 1$ such that
$$R(i) = r_i$$
for each i.
3. Compute $R(n + 1)$ and broadcast
$$c = s + R(n + 1).$$
4. Find the pseudo-dimension k and a pseudo-basis $\mathcal{B} = \{Y_{j1}, \cdots, Y_{jk}\}$ of $\{Y_1, \cdots, Y_n\}$ by using the algorithm of Fig.2.
   Broadcast $k, \mathcal{B}$ and $\Lambda_{\mathcal{B}} = \{j_1, \cdots, j_k\}$.
5. Broadcast $d_u(U_i, Y_i)$ and $d_I(U_i, Y_i)$ for each i.

Step 3. The receiver does the following.

1. Construct FORGED of eq.(4) from $\{E_i = Y_i - X_i \mid i \in \Lambda_{\mathcal{B}}\}$.
2. For each i, find a polynomial $u_i(x)$ with $\deg u_i(x) \leq t$ such that
$$u_i(j) = u_{ij} \text{ for all } j \in d_I(U_i, Y_i),$$
$$u_i(j) = f_i(j) \text{ for all } j \text{ such that } j \notin d_I(U_i, Y_i) \text{ and } j \notin \mathsf{FORGED}$$
3. Find a polynomial $R'(x)$ with $\deg R'(x) \leq n - 1$ such that
$$R'(i) = u_i(i)$$
for each i.[5]
4. Compute $R'(n + 1)$ and output
$$s' = c - R'(n + 1).$$

[4] For simplicity, we assume that there are no such channels in what follows.
[5] "For each i" can be replaced by "for each $i \notin \Lambda_{\mathcal{B}}$" at step 2-2 and step 3-3.

4.3 Security

We first prove the perfect privacy.

Lemma 1. *There is at least one r_i on which the adversary has no information.*

Proof. Consider a non-corrupted channel i such that $i \notin \Lambda_{\mathcal{B}}$. First the sender does not broadcast r_i at step 2-4 because $i \notin \Lambda_{\mathcal{B}}$. Next because $f_i(i)$ is sent through channel i that the adversary does not corrupt, we have

$$r_i = u_{ii} = f_i(i).$$

Further the adversary observes at most t values of $(f_i(1), \cdots, f_i(n))$. Hence the adversary has no information on $r_i = f_i(i)$ because $\deg f_i(x) \leq t$.

Finally there exists at least one non-corrupted channel i such that $i \notin \Lambda_{\mathcal{B}}$ because

$$n - t - |\Lambda_{\mathcal{B}}| \geq n - 2t = 1.$$

□

Therefore, the adversary has no information on $R(n+1)$ from Sec.4.1. Hence she learns no information on s from $c = s + R(n+1)$.

We next prove the perfect reliability. If $j \notin \mathsf{FORGED}$ and $j \notin d_I(U_i, Y_i)$, then $f_i(j) = y_{ij} = u_{ij}$ from the definition of $d_I(U_i, Y_i)$. Therefore, at step 3-2,

$$u_i(j) = u_{ij}$$

for all $j \in d_I(U_i, Y_i)$, and for all j such that $j \notin d_I(U_i, Y_i)$ and $j \notin \mathsf{FORGED}$. This means that $u_i(j) = u_{ij}$ for each $j \in (\overline{\mathsf{FORGED}} \cup d_I(U_i, Y_i))$, where

$$|\overline{\mathsf{FORGED}} \cup d_I(U_i, Y_i))| \geq |\overline{\mathsf{FORGED}}| \geq n - t = (2t+1) - t = t + 1.$$

Further since $\deg u_i(x) \leq t$ and $U_i \in \mathcal{C}$, it holds that

$$(u_i(1), \cdots, u_i(n)) = (u_{i1}, \cdots, u_{in}).$$

In particular, $u_i(i) = u_{ii}$. Therefore from eq.(6), we have that

$$R(i) = r_i = u_{ii} = u_i(i) = R'(i)$$

for each i. Hence we obtain that $R'(x) = R(x)$ because $\deg R'(x) \leq n - 1$ and $\deg R(x) \leq n - 1$. Consequently,

$$s' = c - R'(n+1) = c - R(n+1) = s.$$

Thus the receiver can compute $s' = s$ correctly.

4.4 Efficiency

Let $\mathsf{COM(i)}$ denote the communication complexity of Step i for $i = 1, 2$. Note that $|d_u(U_i, Y_i)| = |d_I(U_i, Y_i)| \leq t$ for each i. Then

$$\begin{aligned}\mathsf{COM}(1) &= O(n(n+n)|\mathsf{F}|) = O(n^2|\mathsf{F}|),\\ \mathsf{COM}(2) &= O((|d_I(U_i,Y_i)|\log_2 n + |d_u(U_i,Y_i)||\mathsf{F}|)n^2\\ &\quad +(\log_2 n + n|\mathcal{B}||\mathsf{F}| + |\Lambda_{\mathcal{B}}|\log_2 n)n + |\mathsf{F}|n)\\ &= O(tn^2\log_2 n + tn^2|\mathsf{F}| + n\log_2 n + n^2t|\mathsf{F}| + tn\log_2 n + |\mathsf{F}|n)\\ &= O(n^2t|\mathsf{F}|)\end{aligned}$$

because $|\mathcal{B}| = |\Lambda_{\mathcal{B}}| \le t$. Hence the total communication complexity is $O(n^2t|\mathsf{F}|)$. The transmission rate is $O(n^2t)$ because the sender sends one secret.

It is easy to see that the computational cost of the sender and the receiver are polynomial in n.

5 More Efficient 2-Round Protocol

In our basic 2-round protocol, the transmission rate was $O(n^2t)$. In this section, we reduce it to $O(n^2)$. We will use nt codewords $X_i \in \mathcal{C}$ to send t^2 secrets in this section while n codewords were used to send a single secret in the basic 2-round PSMT.

5.1 Protocol

The sender wishes to send $\ell = t^2$ secrets $s_1, s_2, \ldots, s_\ell \in \mathsf{F}$ to the receiver.

Step 1. The receiver does the following for each channel i.
For $h = 0, 1, \cdots, t-1$;

1. He chooses a random polynomial $f_{i+hn}(x)$ such that $\deg f_{i+hn}(x) \le t$.
2. He sends
$$X_{i+hn} = (f_{i+hn}(1), \cdots, f_{i+hn}(n))$$
through channel i, and the sender receives
$$U_{i+hn} = (u_{i+hn,1}, \cdots, u_{i+hn,n})$$
3. Through each channel j, he sends $f_{i+hn}(j)$ and the sender receives
$$y_{i+hn,j} = f_{i+hn}(j) + e_{i+hn,j},$$
where $e_{i+hn,j}$ is the error caused by the adversary. Let
$$Y_{i+hn} = (y_{i+hn,1}, \cdots, y_{i+hn,n}),\ E_{i+hn} = (e_{i+hn,1}, \cdots, e_{i+hn,n}).$$

Step 2. The sender does the following.

1. Find the pseudo-dimension k and a pseudo-basis $\mathcal{B} = \{Y_{j1}, \ldots, Y_{jk}\}$ of $\{Y_1, \cdots, Y_{tn}\}$ by using the algorithm of Fig.2.
Broadcast $k, \mathcal{B}$ and $\Lambda_{\mathcal{B}} = \{j_1, \cdots, j_k\}$.

2. For $i = 1, \cdots, n$,
   (a) If $u_{i+hn,i} \neq y_{i+hn,i}$ or $|d_u(U_{i+hn}, Y_{i+hn})| \geq k + 1$[6] or $U_{i+hn} \notin \mathcal{C}$ for some h, then broadcast "ignore channel i".[7] This channel will be ignored from now on because it is forged clearly.
   (b) Else define r_{i+hn} as

$$r_{i+hn} = u_{i+hn,i} = y_{i+hn,i} \tag{7}$$

   for $h = 0, \cdots, t-1$.
3. Find a polynomial $R(x)$ with $\deg R(x) \leq nt - 1$ such that

$$R(i + hn) = r_{i+hn}$$

   for each $i + hn$.
4. Compute $R(nt + 1), \cdots, R(nt + \ell)$ and broadcast

$$c_1 = s_1 + R(nt + 1), \cdots, c_\ell = s_\ell + R(nt + \ell).$$

5. Broadcast $d_u(U_{i+hn}, Y_{i+hn})$ and $d_I(U_{i+hn}, Y_{i+hn})$ for each $i + hn$.

Step 3. The receiver does the following.

1. Construct FORGED of eq.(4) from $\{E_i = Y_i - X_i \mid i \in \Lambda_{\mathcal{B}}\}$.
2. For each $i + hn$, find a polynomial $u_{i+hn}(x)$ with $\deg u_{i+hn}(x) \leq t$ such that

$$u_{i+hn}(j) = u_{i+hn,j} \text{ for all } j \in d_I(U_{i+hn}, Y_{i+hn})$$
$$u_{i+hn}(j) = f_{i+hn}(j) \text{ for all } j \text{ such that } j \notin d_I(U_{i+hn}, Y_{i+hn}) \text{ and } j \notin \mathsf{FORGED}$$

3. Find a polynomial $R'(x)$ with $\deg R'(x) \leq nt - 1$ such that

$$R'(i + hn) = u_{i+hn}(i)$$

   for each $i + hn$.[8]
4. Compute $R'(nt + 1), \cdots, R'(nt + \ell)$ and output

$$s'_1 = c_1 - R'(nt + 1), \cdots, s'_\ell = c_\ell - R'(nt + \ell).$$

5.2 Security

We first prove the perfect privacy.

Lemma 2. *There exists a subset $A \subset \{r_1, \cdots, r_{tn}\}$ such that $|A| \geq \ell$ and the adversary has no information on A.*

[6] k is the number of channels that the adversary forged on $\{Y_{i+hn}\}$.

[7] For simplicity, we assume that there are no such channels in what follows.

[8] "For each $i + hn$" can be replaced by "for each $i + hn \notin \Lambda_{\mathcal{B}}$" at step 2-3 and step 3-3.

Proof. Consider a non-corrupted channel i such that $i + hn \notin \Lambda_{\mathcal{B}}$. First the sender does not broadcast r_{i+hn} at step 2-1 because $i + hn \notin \Lambda_{\mathcal{B}}$. Next since $f_{i+hn}(i)$ is sent through channel i that the adversary does not corrupt, we have

$$r_{i+hn} = u_{i+hn,i} = f_{i+hn}(i).$$

Further the adversary observes at most t values of $(f_{i+hn}(1), \cdots, f_{i+hn}(n))$. Hence the adversary has no information on $r_{i+hn} = f_{i+hn}(i)$ because $\deg f_{i+hn}(x) \le t$.

Note that the adversary corrupts at most t channels and for each corrupted channel i, the adversary gets $r_i, r_{i+n}, \ldots, r_{i+(t-1)n}$. Therefore, there exists a subset $A \subset \{r_1, \cdots, r_{tn}\}$ such that

$$|A| \ge nt - |\Lambda_{\mathcal{B}}| - t^2 = nt - k - t^2$$

and the adversary has no information on A. Finally

$$nt - k - t^2 \ge (2t+1)t - t - t^2 = t^2 = \ell.$$

□

Therefore, the adversary has no information on $R(nt + 1)$, …, $R(nt + \ell)$ from Sec.4.1. Hence she learns no information on s_i for $i = 1, \cdots, \ell$.

We next prove the perfect reliability. If $j \notin \mathsf{FORGED}$ and $j \notin d_I(U_{i+hn}, Y_{i+hn})$, then $f_{i+hn}(j) = y_{i+hn,j} = u_{i+hn,j}$ from the definition of $d_I(U_{i+hn}, Y_{i+hn})$. Therefore,

$$u_{i+hn}(j) = u_{i+hn,j}$$

for all $j \in d_I(U_{i+hn}, Y_{i+hn})$, and for all j such that $j \notin d_I(U_{i+hn}, Y_{i+hn})$ and $j \notin \mathsf{FORGED}$. This means that $u_{i+hn}(j) = u_{i+hn,j}$ for each $j \in (\overline{\mathsf{FORGED}} \cup d_I(U_{i+hn}, Y_{i+hn}))$, where

$$|\overline{\mathsf{FORGED}} \cup d_I(U_{i+hn}, Y_{i+hn}))| \ge |\overline{\mathsf{FORGED}}| \ge n - t = 2t + 1 - t = t + 1.$$

Further since $\deg u_{i+hn}(x) \le t$ and $U_{i+hn} \in \mathcal{C}$, it holds that

$$(u_{i+hn}(1), \cdots, u_{i+hn}(n)) = (u_{i+hn,1}, \cdots, u_{i+hn,n}).$$

In particular, $u_{i+hn}(i) = u_{i+hn,i}$. Therefore from eq.(7), we have that

$$R(i + hn) = r_{i+hn} = u_{i+hn,i} = u_{i+hn}(i) = R'(i + hn)$$

for each $i + hn$. Hence we obtain that $R'(x) = R(x)$ because $\deg R'(x) \le nt - 1$ and $\deg R(x) \le nt - 1$. Consequently,

$$s'_i = c_i - R'(nt + i) = c_i - R(nt + i) = s_i.$$

Thus the receiver can compute $s'_i = s_i$ correctly for $i = 1, \cdots, \ell$.

5.3 Efficiency

Let $\mathsf{COM(i)}$ denote the communication complexity of Step i for $i = 1, 2$. Note that $|d_u(U_{i+hn}, Y_{i+hn})| = |d_I(U_{i+hn}, Y_{i+hn})| \le t$ for each $i + hn$. Then

$$\begin{aligned}\mathsf{COM}(1) &= O(tn(n+n)|\mathsf{F}|) = O(tn^2|\mathsf{F}|),\\ \mathsf{COM}(2) &= O((|d_I(U_{i+hn}, Y_{i+hn})| \log_2 n + |d_u(U_{i+hn}, Y_{i+hn})||\mathsf{F}|)tn \times n \\ &\qquad + (\log_2 n + n|\mathcal{B}||\mathsf{F}| + |\Lambda_{\mathcal{B}}| \log_2 n)n + t^2|\mathsf{F}|n) \\ &= O(n^2t^2 \log_2 n + n^2t^2|\mathsf{F}| + n \log_2 n + n^2t|\mathsf{F}| + tn \log_2 n + t^2|\mathsf{F}|n) \\ &= O(n^2t^2|\mathsf{F}|)\end{aligned}$$

because $|\mathcal{B}| = |\Lambda_{\mathcal{B}}| \leq t$. Hence, the total communication complexity is $O(n^2t^2|\mathsf{F}|)$, and the transmission rate is $O(n^2)$ because the sender sends t^2 secrets.

It is easy to see that the computational costs of the sender and the receiver are both polynomial in n.

6 Final 2-Round PSMT

The transmission rate is still $O(n^2)$ in the 2-round PSMT shown in Sec.5. In this section, we show how to reduce it to $O(n)$ by using the technique of [1, page 406] and [8]. Then we can obtain the first 2-round PSMT for $n = 2t + 1$ such that not only the transmission rate is $O(n)$ but also the computational costs of the sender and the receiver are both polynomial in n.

6.1 Generalized Broadcast

Suppose that the receiver knows the locations of k ($\leq t$) channels that the adversary forged. For example, suppose that the receiver knows that channels $1, 2, \cdots, k$ are forged by the adversary. Then the adversary can corrupt at most $t - k$ channels among the remaining $n - k$ channels $k + 1, \cdots, n$. In this case, it is well known that the sender can send $k + 1$ field elements $u_1, u_2, \ldots, u_{k+1}$ reliably with the communication complexity $O(n|\mathsf{F}|)$ as shown below.

1. The sender finds a polynomial $p(x)$ with $\deg p(x) \leq k$ such that $p(1) = u_1$, $p(2) = u_2, \ldots, p(k + 1) = u_{k+1}$.
2. She sends $p(i)$ through channel i for $i = 1, \cdots, n$.

Without loss of generality, suppose that the receiver knows that channels $1, \cdots, k$ are forged by the adversary. Then he consider a shortened code such that a codeword is $(p(k+1), \cdots, p(n))$. The minimum Hamming distance of this code is $(n - k) - k = 2t + 1 - 2k = 2(t - k) + 1$. Hence the receiver can correct the remaining $t - k$ errors.

This means that the receiver can decode $(p(k + 1), \cdots, p(n))$ correctly. Then he can reconstruct $p(x)$ by using Lagrange formula because

$$n - k = 2t + 1 - k \geq 2k + 1 - k = k + 1 \geq \deg p(x) + 1.$$

Therefore he can obtain $u_1 = p(1), \ldots, u_{k+1} = p(k + 1)$ correctly.

6.2 How to Improve Step 2-5

Step 2-5 is the most expensive part in the 2-round PSMT shown in Sec.5. In this subsection, we will show a method which reduces the communication complexity of step 2-5 from $O(n^2t^2|\mathsf{F}|)$ to $O(n^2t|\mathsf{F}|)$.

At step 2-5, the sender *broadcasts*

$$d_u(U_{i+hn}, Y_{i+hn}) \text{ and } d_I(U_{i+hn}, Y_{i+hn})$$

for each $i + hn$. Note that the size of all $d_u(U_{i+hn}, Y_{i+hn})$ is bounded by

$$\sum_{i=1}^{n} \sum_{h=0}^{t-1} |d_u(U_{i+hn}, Y_{i+hn})| \leq knt \tag{8}$$

because $|d_u(U_{i+hn}, Y_{i+hn})| \leq k$ from Step 2-2(a), where

$$k = |\mathcal{B}| = |\mathsf{FORGED}|$$

is the number of channels that the adversary forged on $\{Y_{i+hn}\}$. On the other hand, the following lemma holds.

Lemma 3. *The sender can send $k + 1$ field elements reliably with the communication complexity $O(n|\mathsf{F}|)$ at step 2-5.*

Proof. The sender knows the value of k because she computes $\mathcal{B}$. The receiver knows the locations of k forged channels because he computes FORGED. Therefore, we can use the generalized broadcasting technique shown in Sec.6.1. □

Now from eq.(8) and Lemma 3, it is easy to see that the communication complexity of step 2-5 can be reduced to $O(n^2 t|\mathsf{F}|)$.

6.3 Final Efficiency

Consequently, we obtain $\mathsf{COM}(2) = O(n^2 t|\mathsf{F}|)$ because the communication complexity of step 2-5 is now reduced to $O(n^2 t|\mathsf{F}|)$. On the other hand, $\mathsf{COM}(1) = O(n^2 t|\mathsf{F}|)$ from Sec.5.3. To summarize,

$$\mathsf{COM}(1) = O(n^2 t|\mathsf{F}|) \text{ and } \mathsf{COM}(2) = O(n^2 t|\mathsf{F}|)$$

in our final 2-round PSMT. Hence, the total communication complexity is $O(n^3|\mathsf{F}|)$ because $n = 2t + 1$.

Now the transmission rate is $O(n)$ because the sender sends t^2 secrets which is $O(n^2|\mathsf{F}|)$. Finally, it is easy to see that the computational costs of the sender and the receiver are both polynomial in n.

References

1. Agarwal, S., Cramer, R., de Haan, R.: Asymptotically Optimal Two-Round Perfectly Secure Message Transmission. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 394–408. Springer, Heidelberg (2006)
2. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly Secure Message Transmission. J. ACM 40(1), 17–47 (1993)

3. Desmedt, Y., Wang, Y., Burmester, M.: A Complete Characterization of Tolerable Adversary Structures for Secure Point-to-Point Transmissions Without Feedback. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 277–287. Springer, Heidelberg (2005)
4. Fitzi, M., Franklin, M., Garay, J., Vardhan, S.: Towards Optimal and Efficient Perfectly Secure Message Transmission. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 311–322. Springer, Heidelberg (2007)
5. Hirt, M., Maurer, U.: Player Simulation and General Adversary Structures in Perfect Multiparty Computation. J. Cryptology 13(1), 31–60 (2000)
6. Kumar, M.V.N.A., Goundan, P.R., Srinathan, K., Rangan, C.P.: On perfectly secure communication over arbitrary networks. In: PODC 2002, pp. 193–202 (2002)
7. Md Sayeed, H., Abu-Amara, H.: Efficient Perfectly Secure Message Transmission in Synchronous Networks. Inf. Comput. 126(1), 53–61 (1996)
8. Srinathan, K., Narayanan, A., Rangan, C.P.: Optimal Perfectly Secure Message Transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004)

Protocols and Lower Bounds for Failure Localization in the Internet

Boaz Barak, Sharon Goldberg, and David Xiao

Princeton University, Princeton, NJ 08544

Abstract. A secure *failure-localization path-quality-monitoring* (FL-PQM) protocols allows a sender to localize faulty links on a single path through a network to a receiver, even when intermediate nodes on the path behave adversarially. Such protocols were proposed as tools that enable Internet service providers to select high-performance paths through the Internet, or to enforce contractual obligations. We give the first formal definitions of security for FL-PQM protocols and construct:

1. A simple FL-PQM protocol that can localize a faulty link *every time* a packet is not correctly delivered. This protocol's communication overhead is $O(1)$ additional messages of length $O(n)$ per packet (where n is the security parameter).
2. A more efficient FL-PQM protocol that can localize a faulty link when a *noticeable fraction* of the packets sent during some time period are not correctly delivered. The number of additional messages is an arbitrarily small fraction of the total number of packets.

We also prove lower bounds for such protocols:

1. Every secure FL-PQM protocol requires *each* intermediate node on the path to have some shared secret information (*e.g.* keys).
2. If secure FL-PQM protocols exist then so do one-way functions.
3. Every *black-box* construction of a FL-PQM protocol from a random oracle that securely localizes every packet and adds at most $O(\log n)$ messages overhead per packet requires *each* intermediate node to invoke the oracle.

These results show that implementing FL-PQM requires active cooperation (*i.e.* maintaining keys and agreeing on, and performing, cryptographic protocols) from *all* of the intermediate nodes along the path. This may be problematic in the Internet, where links operate at extremely high speeds, and intermediate nodes are owned by competing business entities with little incentive to cooperate.

Keywords: Failure localization, secure routing, black-box separation.

1 Introduction

The Internet is an indispensable part of our society, and yet its basic foundations remain vulnerable to attack. Secure routing protocols seek to remedy this by not only providing guarantees on the correct setup of paths from sender to receiver through a network (*e.g.* secure BGP [16]), but also by verifying that

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 341–360, 2008.

data packets are actually delivered correctly along these paths. Packet delivery is surprisingly susceptible to simple attacks; in the current Internet, packets are typically sent along a single path from sender to receiver, and so a malicious node along the data path can easily drop or modify packets before they reach their destination. To detect and respond to such attacks, the networking community has recently been studying monitoring and measurement protocols that are used to obtain information about packet loss events on a data path (*e.g.* [2,3,4,5,7,18,19,21,23,24]). The motivation for such protocols is twofold. First, they provide the sender with information that he can use during path setup to select a single, high-performance path to the receiver from the multiple available paths through the network [11]. Second, since Internet service is a contractual business, where senders pay nodes along the data path to carry their packets, information from Internet measurement protocols is highly valuable for enforcing contractual obligations between nodes. In fact, Laskowski and Chuang [17] recently argued that this information is not only valuable, but also *necessary* to counter the Internet industry's growing trend towards degraded path performance. Note that if monitoring protocols are used to enforce contractual obligations, nodes may have an economic incentive to bias the information obtained from these protocols.

In this work we provide a rigorous cryptographic examination of *secure* monitoring protocols that are robust even in the presence of malicious nodes on the data path. In particular, we study techniques that allow a sender to *localize* the specific links along the data path where packets were dropped or modified— a task that we call *failure-localization path-quality monitoring*. While some protocols for this task are deployed in the Internet today (*e.g.* traceroute [1]), they are not robust to nodes that behave adversarially in order to bias measurements.

1.1 Our Results

We make the following contributions to the study of secure failure-localization path-quality monitoring protocols (in the rest of the paper we call these simply *failure localization* or FL protocols). Throughout the paper, we use the word "packet" to denote data that the sender wishes to transmit, and "message" to refer to both data packets and FL-protocol-related messages.

Definition. In Section 2, we give the first formal definition of security for failure localization protocols. We note that some of the previous FL protocols suggested in the literature, such as [21,4,2], do *not* satisfy our definition. (We sketch attacks in Appendix A.)

We give two variants of the definition— *per-packet* security requires localizing a link each time a packet is not delivered, while *statistical* security only requires this when a noticeable fraction of packets fail to arrive. An important feature of our definition is that it accounts for the fact that messages can be dropped in the Internet for benign reasons like congestion. We note that care must be taken to design protocols that are simultaneously robust to both adversarial behaviour and benign congestion. We discuss the effect of this assumption on some previous work [4] in Appendix A.

Protocols. We present two simple protocols satisfying our per-packet (Section 3.1) and statistical (Section 3.2) security definitions. Both of these protocols do not modify the packets sent on the path; instead, they add additional messages. Thus our protocols have the important advantage of allowing backwards compatibility with the current techniques for processing packets in a router, minimizing latency in the router, and not increasing packet size.

Our main measure of efficiency for such protocols is communication overhead—the number and size of messages added by the protocols. The per-packet protocol adds a single $O(n)$-length message to every packet sent (n is the security parameter), and $O(K)$ additional $O(n)$-length messages when a failure occurs (where K is the number of nodes on the path). The statistical protocol only needs $O(K)$ additional $O(n)$-length messages per T packets sent. In our setting K is constant, while T could be $\mathrm{poly}(n)$, which implies the statements in the abstract.

Lower bounds. Like many of the protocols in the literature [3, 4, 21, 19, 2], both of our protocols require cryptographic keys and computations at each node. These requirements are considered severe in the networking literature; setting up a key infrastructure and agreeing on cryptographic primitives is challenging in the distributed world of the Internet, where each node is owned by a different entity with sometimes incompatible incentives. However, in Section 4 we show that these requirements are to some degree *inherent* by:

1. Proving that every secure (per-packet or statistical) FL protocol requires a key infrastructure, or more precisely, that intermediate nodes and Alice and Bob must all share some secret information between each other.
2. Proving that a one-way function can be constructed from any secure FL protocol.
3. Giving evidence that any practical per-packet secure FL protocol must use these keys in a cryptographic way at *every node* (*e.g.* , it does not suffice to use the secret information with some simple, non-cryptographic, hash functions as in [7]). We show that in every black-box construction of such a protocol from a random oracle, where at most $O(\log n)$ protocol messages are added per packet, then every intermediate node must query the random oracle. We note that known protocols designed for Internet routers currently avoid using public-key operations, non-black-box constructions, or adding more than a constant number of protocol messages per packet. We also show that for statistically-secure FL, or FL protocols adding $\omega(\log n)$ messages per packet, the necessity of cryptography depends on subtle variations in the security definition.

Implications of our results. Our lower bounds raise questions about the practicality of deploying FL protocols. In small highly-secure networks or for certain classes of traffic, the high key-management and cryptographic overhead required for FL protocols may be tolerable. However, FL protocols may be impractical for widespread deployment in the Internet; firstly because intermediate nodes are owned by competing business entities that may have little incentive to set up a key infrastructure and agree on cryptographic protocols, and secondly because cryptographic computations are expensive in the core of the Internet, where packets must be processed at extremely high speeds (about 2 ns per packet). Thus, our work can be seen as a motivation for finding security functionalities for the Internet that are more practical than failure localization.

1.2 Related Work

Some of this work (in particular, the results of Section 3 and a weaker version of Theorem 5) appeared in our earlier technical report [8]. We built on [8] in [9], where, together with Jennifer Rexford and Eran Tromer, we gave formal definitions, constructions, and lower bounds for the simpler task of *path-quality monitoring* (PQM). In a PQM protocol the sender only wishes to *detect* if a failure occurred, rather than localize the specific faulty link along the path. We use the results from [9,8] in Section 3.2 to show how a PQM protocol can be composed to obtain a statistical FL protocol, and in Section 4.2 to argue that FL protocols need cryptographic computations.

In addition to the FL protocols from the networking literature [3,4,21,19,2,24], our work is also related to the work on secure message transmission (SMT) begun by Dolev, Dwork, Waart, and Yung in [6]. In SMT, a sender and receiver are connected by a multiple parallel wires, any of which can be corrupted by an adversary. Here, we consider a single path with a series of nodes that can be corrupted by an adversary, instead of multiple parallel paths. Furthermore, while multiple parallel paths allow SMT protocols to *prevent* failures, in our single path setting, an adversarial intermediate node can always block the communication between sender and receiver. As such, here we only consider techniques for *detecting and localizing* failures.

2 Our Model

In a failure localization (FL) protocol, a sender Alice wants to know whether the packets she sends to receiver Bob arrive unmodified, and if not, to find the link along the path where the failure occurred (see Figure 1). We say a *failure* or *fault* occurs when a data packet that was sent by Alice fails to arrive unmodified at Bob. Following the literature, we assume that Alice knows the identities of all the nodes of the data path. We work in the setting where all traffic travels on symmetric paths (*i.e.* intermediate nodes have bi-directional communication links with their neighbors, and messages that sender Alice sends to receiver Bob traverse the same path as the messages that Bob sends back to Alice). We say that messages travelling towards Alice are going *upstream*, and messages travelling towards Bob are going *downstream*. An adversary Eve can occupy any set of nodes on the path between Alice and Bob, and can add, drop, or modify messages sent on the links adjacent to any of the nodes she controls. She can also use timing information to attack the protocol.

Localizing links, not nodes. It is well known that an FL protocol can only pinpoint a *link* where a failure occurred, rather than the *node* responsible for the failure. To see why, refer to Figure 1, and suppose that (a) Eve controlling node R_2 becomes unresponsive by ignoring all the messages she receives from R_1. Now suppose that (b) Eve controls node R_1 and pretends that R_2 is unresponsive by dropping all communication to and from R_2. Because cases (a) and (b) are completely indistinguishable from Alice's point of view, at best Alice can localize the failure to link $(1, 2)$.

R_A (Alice) $\rightleftarrows$ R_1 $\rightleftarrows$ R_2 $\rightleftarrows \ldots \rightleftarrows$ R_K $\rightleftarrows$ R_B (Bob)

Fig. 1. A path from Alice to Bob via K intermediate nodes

Congestion. Congestion-related packet loss is widespread on the current Internet, caused by protocols like TCP [15] that naturally drive the network into a state of congestion. Our definition accounts for congestion by assuming links can drop each message independently with some probability. One could come up with other models for congestion (*e.g.* allowing Eve to specify the distribution of congestion-related packet loss), and for some plausible choices our positive results will still hold. However, we use independent drops for the sake of simplicity. Furthermore, assuming that congestion is not controlled by the adversary only strengthens our negative results and makes our model more realistic.

2.1 Security Definition

Let n be the security parameter. A failure localization protocol consists of an efficient initialization algorithm Init taking n uniformly random bits and generating keys for each node, and efficient node algorithms $\mathsf{Alice}, \mathsf{Bob}, R_1, \ldots, R_K$ which take in a key and communicate with each other as in Figure 1. We always fix $K = O(1)$ independent of n.[1] The Alice algorithm takes in a packet that she wants to send to Bob. If communication is successful, then the Bob algorithm outputs the packet that Alice sent. Our security definitions are game-based:

Definition 1 (Security game for FL). The game begins when Eve chooses a subset of nodes $E \subseteq \{1, \ldots, K\}$ that she will occupy for the duration of the game. The Init algorithm is then used to generate keys for each node, and Eve is given the keys for the nodes $i \in E$ that she controls. We define an oracle Source that generates data packets d for the Alice algorithm to send. We allow Eve to choose the packets that the Source oracle generates, subject to the condition that she may not choose the same packet more than once during the game.[2] We

[1] Typically in the Internet, the path length K is less than 20 when nodes represent individual routers, and when nodes represent Internet Service Providers (ISPs) then there are on average $K \approx 4$, and no more 7 nodes on a typical path [16].

[2] We make this assumption because there is natural entropy in packet contents, due to TCP sequence numbers and IP ID fields [7]. To enforce this assumption in practice, protocol messages can be timestamped with with an expiry time, such that with high probability (over the entropy in the packet contents), no repeated packets are sent for the duration of the time interval for which the protocol messages are valid.

allow Eve to add, drop, or modify any of the messages sent on the links adjacent to the nodes she occupies. We include congestion in our model by requiring that, for each message sent on each link on the path, the link *goes down* or drops the message with some constant probability $\rho > 0$. Notice that this means that a failure can happen at links not adjacent to a node occupied by Eve.

We introduce the notion of time into our model by assuming that the game proceeds in discrete timesteps; in each timestep, a node can take in an input and produce an output, and each link can transmit a single message. (Thus, each timestep represents an event occurring on the network.) Because it is expensive to have securely synchronized clocks in a distributed system like the Internet,[3] we do *not* allow the honest algorithms to take timing information as an input. However, to model timing attacks, we assume that Eve knows which timestep that the game is in.

Then, our per-packet security definition uses the the game defined in Definition 1:

Definition 2 (Per-packet security for FL). In the per-packet security game, Eve gets to interact with the Source oracle and the "honest" node algorithms as in Definition 1, until she decides to stop. For each packet sent, Alice must output either $\surd$ (*i.e.* not raise an alarm) or a link ℓ (*i.e.* raise an alarm and localize a failure to ℓ). We assume that the game is *sequential:* Alice must output a decision for each data packet before starting to transmit the next data packet (see remarks below). We say that an FL protocol is *per-packet secure* if the following hold:

1. (*Secure localization*). For every packet d sent by the Source oracle that is not successfully output by Bob, then Alice outputs a link ℓ such that either (a) link ℓ is adjacent to a node occupied by Eve, or (b) link ℓ went down due to congestion for one of the messages (including FL protocol messages) associated with sending packet d from Alice to Bob.
2. (*No false positives*). For every packet d sent by the Source oracle that is successfully output by Bob, for which there was no congestion, and for which Eve does not deviate from the protocol, Alice outputs $\surd$.

We need to introduce a few new concepts for our statistical security definition. First, we define an *interval* as a sequence of T packets (and associated FL protocol messages) that Alice sends to Bob.[4] Next, we use the following parameters: a false alarm threshold α, a detection threshold for the path β (where $0 < \alpha < \beta < 1$) and an error parameter $\delta \in \{0, 1\}$. Usually, we will set α such that congestion alone almost never causes the failure rate on a path to exceed the false alarm threshold.

Definition 3 ((α, β, δ)-Statistical security for FL). In the statistical security game, Eve is allowed to choose the number of *intervals* for which she wants

[3] Indeed, the NTP protocol used for clock synchronization on the Internet is not secure [12], and thus should not be used as an input to a secure FL protocol.

[4] We can think of an interval as all the packets sent in some time period (*e.g.* approximately 10^7 packets are sent 100 msec over a 5 Gbps Internet path).

to interact with the Source oracle and the honest nodes as in Definition 1. The number of packets per interval T may grow with n, but is always at least some minimum number depending α, β, δ, K. At the end of each *interval*, Alice needs to output either $\surd$ (*i.e.* not raise an alarm) or a link ℓ (*i.e.* raise an alarm and localize a link). The game is sequential; Alice must output a decision for each interval before starting the next interval. Then, an FL protocol is *statistically secure* if the following hold:

1. (*Secure localization*). For any interval in the security game where Eve causes the failure rate on the path to exceed the detection threshold β, then with probability $1-\delta$ Alice raises alarm for a link ℓ that is adjacent to Eve, or a link ℓ whose failure rate exceeds $\frac{\alpha}{K+1}$.
2. (*Few false positives*). For any interval in the security game where Eve does not deviate from the correct algorithm R_i of any of the nodes $i \in E$ that she controls and the failure rate on each link is below the (per-link) false alarm threshold $\frac{\alpha}{K+1}$, then the probability that Alice outputs $\surd$ is at least $1-\delta$.

We now discuss some properties of our security definition.

Benign and malicious failures. Our security definitions require Alice to accurately localize failures, but these failures may be caused by Eve, or may be the result of *benign causes*, such as congestion. We do not require Alice to distinguish between benign or malicious (*i.e.* due to Eve) failures, because Eve can always drop packets in a way that "looks like" congestion.

Sequential games. For simplicity, in our per-packet security game we required Alice to make FL decisions before she sends a new data packet. This is to capture the fact that such protocols should provide "real-time" information about the quality of the paths she uses, and so we did not allow Alice in the per-packet case to make decisions only after sending many packets (as is done in the statistical security case). We note that while our negative results (*i.e.* attacks) are sequential, our positive results (*i.e.* , protocols) do not use the assumption of sequential execution in any way, and are secure in a more general setting where Eve can choose when Alice needs to output an FL decision each packet. We emphasize that the sequential assumption does *not* prevent Alice from keeping state and using information from *past* packets in order to make FL decisions. (Though none of our positive results require that Alice does this.)

Movements of the adversary. Our model does not allow Eve to move from node to node in a single security game. This assumption makes sense when Eve models a Internet service provider that tries, for business reasons, to bias the results of FL protocol. Furthermore, when Eve is an external attacker or virus that compromises a router, "leaving" a router means that the legitimate owner of the router removed the attacker from the router, *e.g.* by refreshing its keys. We model this key refresh process as a re-start of the security game. Furthermore, in practice "movements" to a new router happen infrequently, since an external attacker typically needs a different strategy each time it compromises a router owned by a different business entity.

Generalizations. All our results generalize to the setting where congestion rates, false alarm thresholds, and detection thresholds are different per link; we set them all equal here for simplicity. Our negative results also hold for the weaker adversary model where Eve can occupy only one node and the Source oracle generates independent (efficiently-samplable) packets from a distribution that is *not* controlled by Eve.

3 Protocols

We now present protocols for secure per-packet and statistical FL. Our protocols are related, though not identical to those of [2,3,4]. (In Appendix A we show that the protocols in [2,4] do not satisfy our security definitions.)

We use the notation $[m]_k$ to denote a message m authenticated by a key k using a *message authentication code* (MAC); such schemes can be constructed from any one-way function [10,22]. We'll often use the well-known notion of an *onion report*: if every node R_i wants to transmit a report τ_i to Alice in an authenticated way, then we define inductively $\theta_{K+1} = [(K+1, \tau_{\mathsf{Bob}})]_{k_{\mathsf{Bob}}}$ and for $1 \leq i \leq K$, $\theta_i = [(i, \tau_i, \theta_{i+1})]_{k_i}$. That is, each R_i's report is appended with its downstream neighbors' reports before being authenticated and passed upstream. Onion reports prevent Eve from selectively dropping reports — if Eve occupies R_j and wants to drop the report τ_i of R_i for some $i > j$ then, under the assumption that Eve cannot forge MACs, Alice will discover that R_j tampered with the onion report. We also note that every time we send or store a packet d in acknowledgments and reports, we could save space by replacing d with an $O(n)$-length hash of d via some collision-resistant hash function, where n is the security parameter.

3.1 Optimistic Per-Packet FL Protocol

We assume that each node R_i shares a symmetric key k_i with Alice. For each packet that Alice sends, the protocol proceeds in two phases:

The detect phase. Alice stores each packet d that she sends to Bob. When Bob receives the packet d, he responds with an ack of the form $a = [d]_{k_B}$. Alice removes the the packet d from storage when she receives a validly MAC'ed corresponding ack, and raises an alarm if a valid ack is not received.[5] We also require each intermediate node to store each data packet and corresponding ack.

The localize phase. This phase is run only if Alice raises an alarm for a packet d. Alice sends an *onion report request* $q = (\mathsf{report}, d)$ downstream towards Bob. To respond to the request, each node R_i checks if he stored data packet d; if he

[5] In practice, each packet d should be stored along with a local timeout at Alice. If the ack does not arrive before the timeout expires, then Alice should raise an alarm.

did, R_i sets $\tau_i = (q, i, d, a)$ where a is the ack he saw corresponding to packet d, and substituting the symbol $\perp$ for d and/or a if he failed to to receive that packet or an ack. R_i then creates an onion report θ_i using τ_i as described above. In the onion report, R_i can substitute the symbol $\theta_{i+1} = \perp$ if he fails to receive a θ_{i+1} from R_{i+1}.

To localize the failure, Alice classifies the onion reports that she received in response to her onion report request q. An onion report $\theta_i = [q', i', d', a', \theta_{i+1}]_{k_i}$ is "consistent" if it is present, *i.e.* $\theta_i \neq \perp$, and all of the following four conditions hold. Otherwise, an onion report is "inconsistent".

1. $q' = q$ sent out by Alice.
2. The MAC on θ_i is valid.
3. $d' = d$, where d is the packet queried in q.
4. a' is *not* a valid ack for packet d.

Alice localizes then localizes the upstream-most link $(i, i + 1)$ where the onion reports transition from consistent to inconsistent.

Theorem 1. *The optimistic FL protocol is per-packet secure.*

The proof follows via a simple reduction to the security of the MAC, and is defered to the full version. We remark that the detect phase of this protocol requires a large amount of storage and communication overhead at each node. This high overhead makes this protocol impractical for regular Internet traffic; however, it might be useful for specialized highly-secure networks, or for certain classes of traffic *e.g.* network management traffic.

3.2 A Composition Technique for Statistical FL

We now consider statistical security protocols, that apply results from our previous work on statistical PQM [8,9] to obtain statistical FL protocols with much lower overhead. In a statistical PQM protocol, Alice *detects* whenever the average failure rate exceeds a threshold β (but she need not localize a link).

Here we show how to compose the lightweight PQM protocols we presented in [9] to obtain a statistical FL protocol. While it is possible to give a very general composition theorem, for clarity and concreteness in this version we describe only how to compose the simpler symmetric secure sampling (SSS) protocol of [9]. We defer our more general composition result to the full version of this paper. In particular, we can compose the *secure sketch protocol* of [9] that has a communication overhead of only a single $O(\log T + n)$ length packet for every interval, thus yielding the result stated in Section 1.1.

Symmetric Secure Sampling (SSS), a statistical PQM protocol from [9,8]. SSS requires Alice and Bob to securely designate a random p fraction of the data packets that Alice sends to Bob as "probes", and require that Bob send MAC'd acknowledgments for all the probes. We call p the *probe frequency.* To do this, Alice and Bob share a secret $k = (k_1, k_2)$. For each packet d that Alice sends to Bob, they use k_1 to compute a function Probe that determines whether or not a packet d is a probe and should therefore be stored, and acknowledged. To acknowledge a probe, Bob sends Alice an ack $[d]_{k_2}$ that is MAC'ed using k_2. The Probe function is implemented using a pseudorandom function (PRF) f keyed with k_1, that we think of as mapping strings to integers in $[0, 2^{n-1}]$; we define $\mathsf{Probe}_{k_1}(d)$ output "Yes" if $f_{k_1}(d) < p2^n$ and output "No" otherwise. For each interval, Alice stores each probe packet (*i.e.* each packet d such that $\mathsf{Probe}_{k_1}(d)$ =Yes). At the end of the interval, after T packets are sent, Alice computes V, a count of the number of stored (probe) packets for which she failed to receive a valid ack. She computes the average failure rate as $\frac{V}{pT}$.

A composition that does not work. Perhaps the most natural approach to construct a statistical FL protocol is to have Alice run K simultaneous PQM protocols with each of the intermediate nodes, and use the statistics from each protocol to infer behaviour at each link (similar to [21,4,24]). However, we now show that this composition is vulnerable to the following *timing attack*: Suppose a packet d that Alice sends to Bob is ack'd by innocent node R_j with message a. Then, if Eve occupies node R_i for $i < j-1$, she can determine that R_j originated the ack a by counting the timesteps that elapsed between the timestep in which she saw d and timestep in which she saw a. Then, Eve can implicate R_j by selectively dropping every ack that originates at R_j. Notice that this attack results from the structure of this composition, and cannot be prevented even when acks are encrypted.[6] In practice, this attack can be launched when isolated burst of packets triggers a separate burst of acks at each intermediate node.

Composing PQM to statistical FL. We require that every node R_i shares pairwise keys k_i^A, k_i^B with Alice and Bob respectively. Using k_i^B, each intermediate node runs a statistical PQM protocol with Bob with the following modification: whenever Bob decides to send an ack for a packet d to an intermediate node R_i, Bob will (1) always address the ack to Alice and (2) MAC the ack in onion fashion, starting with k^B_{Alice} (on the inside of the onion) and ending with k_K^B (on the outside of the onion). Each node forwards all acks upstream, and processes only the ack he expects. At the end of the interval u, Alice will send an onion report request $q = (\mathsf{report}, u)$ to all the intermediate nodes. Each intermediate node produces a MAC'd onion report $\theta_i = [q, i, V_i, \theta_{i+1}]_{k_i^A}$ where V_i is his estimate of the average failure rate on the path between himself and Bob. Letting α, β be the false alarm and detection thresholds, when Alice receives the final onion report θ_1, she computes $F_\ell = V_i - V_{i+1}$ for each link $\ell = (i, i+1)$, and outputs ℓ if $F_\ell > \frac{\alpha+\beta}{2(K+1)}$, or if $\ell = (i, i+1)$ is the upstream-most link when the onion report θ_{i+1} refers to the wrong interval, is missing, or is invalidly MAC'ed. We prove that this scheme is secure provided that the interval length T is long enough and the congestion rate ρ is small enough.

Theorem 2. *The composition of SSS described above with probe frequency p satisfies (α, β, δ)-strong statistical security when each interval contains at least $T = O(\frac{K^2}{p(\beta-\alpha)^2} \ln \frac{K}{\delta})$ packets and the congestion rate satisfies $\beta - \alpha \gg K\rho$.*

[6] [24] deals with this by randomizing the sending time of acks.

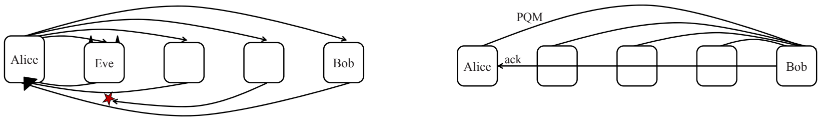


Fig. 2. On the left an insecure composition, on the right our secure composition

Proof. First, observe that the probability that any efficient adversary Eve successfully forges an ack for a dropped packet by forging a MAC used in SSS is negligible. As in the Optimistic Protocol, the probability that any efficient adversary Eve successfully forges the onion report of an honest node (by forging the MAC on the onion report) is negligible as well. Hence, for the rest of this proof assume that Eve does not forge an ack to a dropped packet or validly forge the onion report of an honest node. Moreover, we can assume that Eve does not tamper with the onion report, or else she will implicate a link adjacent to one of the nodes she controls. We now work within a single interval:

- Let V_i be R_i's *estimate* of the failure rate between R_i and Bob.
- Let D_i be a count of the number of packets that were dropped or modified on the path between R_i and Bob.
- Let C_i be the number of acks intended for *any node* that were dropped or modified on the path between Bob and R_i.
- Let $p' = \frac{p}{1-(1-p)^{K+1}}$ be the probability that a node R_i expects an ack to a packet d (*i.e.* $\mathsf{Probe}_{k_i^B}(d) = \text{Yes}$) conditioned on there being at least one node expecting an ack to packet d (*i.e.* $\exists j \in \{0, \dots, K\},\ \mathsf{Probe}_{k_j^B}(d) = \text{Yes}$).[7]

Note that when R_i estimates the average failure rate on the path from R_i to Bob, she is unable to distinguish between dropped packets and dropped acks. Also, it is possible that $D_i > D_{i+1}$ or $C_i > C_{i+1}$ for two adjacent uncorrupted nodes because of congestion. In the absence of adversarial behavior at R_i, the expectation of the estimator V_i that Alice receives in the onion report is $\frac{1}{T}(D_i + \frac{p'}{p}C_i)$. Finally, notice that the average failure rate on link $(i, i{+}1)$ is $\frac{1}{T}(D_i - D_{i+1})$.

Set $\gamma = \frac{\beta-\alpha}{2(K+1)}$. If $T = O(\frac{K^2}{p(\beta-\alpha)^2} \ln \frac{K}{\delta})$ then we have the following lemmata:

Lemma 1 (Deviation of the estimator V_i). *For each $i \notin E$ where E is the set of nodes corrupted by Eve it holds (up to negligible error) that*

$$\Pr\left[\left|V_i - \tfrac{1}{T}(D_i + \tfrac{p'}{p}C_i)\right| > \tfrac{1}{4}\gamma\right] < \tfrac{\delta}{4(K+1)}$$

Lemma 2 (Acks dropped due to congestion). *For each $i, i+1 \notin E$, it holds (up to negligible error) that*

$$\Pr\left[\tfrac{p'}{p}\tfrac{C_i - C_{i+1}}{T} > \tfrac{\gamma}{2}\right] < \tfrac{\delta}{2(K+1)}$$

[7] This quantity is the probability that a node R_i samples an ack that was dropped between R_i and R_B, since at least one node must have sampled the corresponding packet in order for the ack to be transmitted at all.

The proofs of these lemmata are technical, but not difficult. We defer them to the full version. Both proofs are applications of the Chernoff bound under the assumption that the Probe function is implemented with a truly random function; the negligible error refers the difference between a PRF and a truly random function. The proof of Lemma 1 relies on the fact that Eve cannot bias node R_i's estimate of C_i by selectively dropping acks because (1) acks destined for different nodes look identical, and they all originate at Bob (so that an adversary cannot use timing to distinguish between them), and (2) acks are onion MAC'd, so the adversary cannot selectively tamper with an ack intended for an upstream node. The proof of Lemma 2 also relies on the fact that $\beta - \alpha \gg K\rho$.

Few false positives: To prove this, we consider an interval where all the nodes on the path behave honestly, and show that, with probability at least $1-\delta$, Alice will not raise an alarm during this "honest interval".

Consider link $\ell = (i, i+1)$ where the average failure rate is less than the false alarm threshold so $\frac{1}{T}(D_i - D_{i+1}) < \frac{\alpha}{K+1}$. We now show that Alice will not raise an alarm for this link ℓ by proving that Alice's estimate of the failure rate for ℓ, *i.e.* $V_i - V_{i+1}$, does not exceed her alarm decision threshold, *i.e.* $\frac{\alpha+\beta}{2(K+1)}$. We do this by proving that

$$\Pr\left[\left|(V_i - V_{i+1}) - \tfrac{1}{T}(D_i - D_{i+1})\right| > \tfrac{\alpha+\beta}{2(K+1)} - \tfrac{\alpha}{K+1} = \gamma\right] < \tfrac{\delta}{K+1} \qquad (3.1)$$

Notice that "Few false positives" condition follows from (3.1) by a union bound over all $K+1$ links.

To prove (3.1), we start with the expression below, and apply the triangle inequality, and then Lemma 1:

$$\begin{aligned}
&\Pr[|(V_i - V_{i+1}) - (\tfrac{D_i - D_{i+1}}{T} + \tfrac{p'}{p}\tfrac{C_i - C_{i+1}}{T})| > \gamma/2] \\
&\quad \le \Pr[|V_i - \tfrac{1}{T}(D_i + \tfrac{p'}{p}C_i)| > \gamma/4] + \Pr[|V_{i+1} - \tfrac{1}{T}(D_{i+1} + \tfrac{p'}{p}C_{i+1})| > \gamma/4] \\
&\quad \le \tfrac{\delta}{2(K+1)} \qquad (3.2)
\end{aligned}$$

Next, from Lemma 2 we know that $\Pr[\frac{p'}{p}\frac{C_i - C_{i+1}}{T} > \gamma/2] \le \frac{\delta}{2(K+1)}$, and so a union bound over this expression and (3.2) proves (3.1).

Secure localization: We now show that if Eve drops more than a β fraction of packets in any interval, then Alice will catch her with probability at least $1-\delta$. Since the actual failure rate on the path is $\frac{1}{T}D_A > \beta$, we start by applying Lemma 1 to find that Alice's estimate of the failure rate is $V_A > \beta - \frac{\gamma}{4}$ with probability at least $1 - \frac{\delta}{4(K+1)}$. We now use an averaging argument to claim that there exists some link $\ell = (i, i+1)$ such that $V_i - V_{i+1} > \frac{\alpha+\beta}{2(K+1)}$. To see why, suppose for the sake of contradiction that for all i we had $V_i - V_{i+1} \le \frac{\alpha+\beta}{2(K+1)}$. Then, it follows that

$$V_A = \sum_{i=0}^{K}(V_i - V_{i+1}) \le \sum_{\ell} \tfrac{\alpha+\beta}{2(K+1)} = \tfrac{\alpha+\beta}{2} < \beta - \tfrac{\gamma}{4}$$

where $V_{K+1} = 0$ (Bob's estimate of drops to himself is 0). But this contradicts our condition that $V_A > \beta - \frac{\gamma}{4}$, so there is at least one link $\ell = (i, i+1)$ with $V_i - V_{i+1} > \frac{\alpha+\beta}{2(K+1)}$ so that Alice raises an alarm.

Next, recall that we assume that for any link where the true failure rate due to congestion less than $\frac{\alpha}{K+1}$, we have from our proof of the "Few false positives" condition that with probability $\frac{\delta}{K+1}$, Alice does not raise an alarm for link ℓ between two honest nodes. Then, Alice must have raised the alarm for a link adjacent to Eve with probability at least $1-\delta$ (by a union bound) or a link with actual failure rate larger than $\frac{\alpha}{K+1}$, and secure localization follows. ■

4 Lower Bounds

We now argue that in any secure per-packet FL scheme Alice requires shared keys with Bob and the intermediate nodes, and Alice, Bob and each intermediate node must perform cryptographic operations. We only argue for intermediate nodes $R_2, \ldots, R_K$; R_1 is a border case which requires neither keys nor crypto because we assume Alice is always honest.

4.1 Failure Localization Needs Keys at Each Node

Since FL provides strictly stronger security guarantees than path-quality monitoring, it follows from the results in [9] that in any secure FL protocol, Alice and Bob must have shared keys. We also have the following theorem that proves that in any secure FL protocol, *each* intermediate node must share keys with some Alice:

Theorem 3. *Suppose* Init *generates some auxiliary information* aux_i *for each node* R_i *for* $i = 1, ..., K, \mathsf{Alice}, \mathsf{Bob}$. *A FL protocol cannot be (per-packet or statistical) secure if there is any node* $i \in \{2, \ldots, K\}$ *such that* $(\mathsf{aux}_{\mathsf{Alice}}, \mathsf{aux}_1, \ldots, \mathsf{aux}_{i-1})$ *and* aux_i *are independent.*

Proof. Suppose R_i has aux_i that is independent of $(\mathsf{aux}_{\mathsf{Alice}}, \ldots, \mathsf{aux}_{i-1})$. Then, the following two cases are indistinguishable from Alice's view: (a) Node $R_{i+1}$is malicious and blocks communication on link $(i, i+1)$, and (b) Eve occupies node R_{i-1}, and drops packets while simulating case (a) by picking an independent aux'_i and running $R_i(\mathsf{aux}'_i)$ while pretending as if $(i, i+1)$ is down. These two cases are indistinguishable because aux_i is independent of $(\mathsf{aux}_{\mathsf{Alice}}, \ldots, \mathsf{aux}_{i-1})$, and so Alice will localize the failure to the same link in both case (a) and (b). But this breaks security, since R_{i+1}, R_{i-1} do not share a common link. ■

4.2 Failure Localization Needs Crypto at Each Node

In [9], we give a reduction from one-way functions to secure PQM, proving:

Theorem 4 (From [9]). *The existence of a per-packet secure PQM protocol implies the existence of an infinitely-often one-way function (i.o.-OWF).*

Since one-way functions are equivalent to many cryptographic primitives (in the sense that these primitives exist if and only if one-way functions exist [13]), this result can be interpreted to mean that nodes participating in any secure PQM protocol must perform cryptographic computations. Since FL gives a strictly stronger security guarantee than PQM, we also have that in any FL protocol, some node on the data path must perform cryptography. However, Theorem 4 only implies that the *entire system* performs cryptography. We want to prove that any secure FL protocol requires *each intermediate node* $R_1, \dots, R_K$ to perform cryptography. Because it is not clear even how to formalize this in full generality, we instead apply the methodology of Impagliazzo and Rudich [14] to do this for *black-box* constructions of FL protocols from a random oracle RO. We model "performing cryptography" as querying the random oracle, and show that in such a secure FL protocol *each node* must query the RO.

In [14], Impagliazzo and Rudich showed that there can be no secure black-box construction of key agreement (KA) from a random oracle. They argued that if any such KA construction is secure, then it must also be secure in a relativized world where every party has access to a random oracle RO, and a PSPACE oracle. (A PSPACE oracle solves any PSPACE-complete problem, *e.g.* True Quantified Boolean Formulae (TQBF)). Intuitively, in this $(\mathsf{PSPACE}, \mathsf{RO})$ world, every computation is easy to invert *except* for those computed by the RO. They obtain their result by showing, for every possible black-box construction of KA from a random oracle, that there exists an efficient algorithm (relative to $(\mathsf{PSPACE}, \mathsf{RO})$) that breaks the security of KA. Using the the same reasoning, any secure black-box FL protocol constructed from a RO must remain secure even relative to a $(\mathsf{RO}, \mathsf{PSPACE})$ oracle. Then, to obtain our result, it suffices to exhibit an efficient algorithm (relative to $(\mathsf{PSPACE}, \mathsf{RO})$) that breaks security of any black-box FL protocol where one node does not call RO. We do this below.

We will use the notion of an *exchange* to denote a data packet and all the FL-protocol-related messages associated with that packet. Because our game is sequential (see Section 2), Alice's must decide to localize a link ℓ or output $\surd$ before the next exchange begins. We now prove that a per-packet FL protocol with $r = O(\log n)$messages per exchange must invoke the random oracle at every node. We note that protocols where number of messages per packet grows with n are impractical and so "practical" protocols should use $r = O(1)$ messages per exchange. (See Remark 1 below on the possibility of extending this result to statistical security and/or protocols with $\omega(\log n)$ messages per exchange.)

Theorem 5. *Fix a fully black-box per-packet FL protocol that uses access to a random oracle* RO*, where at least one node* R_i *for* $i \in \{2, \dots, I\}$ *never calls the* RO *and where the maximum number of messages per exchange is* $O(\log n)$*. Then there exists an efficient algorithm relative to* $(\mathsf{PSPACE}, \mathsf{RO})$ *that breaks the security of the scheme with non-negligible probability over the randomness of* RO *and the internal randomness of the algorithm.*

The proof of Theorem 5 is quite technical and is deferred to the full version. We sketch the proof, which resembles that of Theorem 3. Eve controls node R_{i-1} and impersonates R_i, but now aux_i is secret, so Eve must first *learn* aux_i:

1. *Learning to impersonate.* Sitting at R_{i-1}, Eve observes t exchanges (t is polynomial in n), where Eve asks Source to transmit a uniformly random data packet. She then uses the learning algorithm of Naor and Rothblum [20] to obtain a pair of impersonator algorithms A', B', whose interaction generates a distribution over transcripts for the $t+1$'th exchange. A' impersonates nodes Alice, $R_1, \ldots, R_{i-1}$ and B' impersonates nodes $R_i, \ldots, R_K$, Bob.
2. *Dropping and impersonating.* On the $t+1$'th exchange, for each message m going from R_{i-1} to R_i, Eve computes a response m' herself using algorithm B' and returns m' to R_{i-1}; she does not send any messages to R_i.

Now, Eve at R_{i-1} will break security if she manages to use B' to impersonate an *honest* exchange during which link $(i, i+1)$ is down. (This breaks security since link $(i, i+1)$ is not adjacent to R_{i-1}.) The crucial observation is that here, Eve need only impersonate node R_i, and that R_i does not "protect" its secret keys by calling the RO. Intuitively, Eve should be able to impersonate R_i since any computations that R_i does are easy to invert in the (PSPACE, RO) world. We now argue that Eve can break security with non-negligible probability.

Recall (Section 2) that Alice is allowed to use information from past exchanges to help her decide how to send messages in new exchanges. Fortunately, the algorithm of Naor and Rothblum [20] is specifically designed to deal with this, and guarantees that observing $t = \text{poly}(n/\varepsilon)$ many exchanges (in Step 1) Eve can obtain, with probability $1-\varepsilon$, algorithms A', B' that generate an impersonated transcript that is ε-statistically close to the "honest" transcript of messages on the link $(i-1, i)$ (generated by interactions of honest Alice, $R_1, ..., R_K$, Bob.)

Suppose Eve obtained an A', B' that satisfy the guarantee above. Our first challenge is that the Naor-Rothblum algorithm does *not* guarantee that A', B' generates an impersonated transcript that is statistically close to the "honest" transcript of messages on $(i-1, i)$ when *the observer has access to the* RO. Fortunately, with probability ρ^r all the messages sent from R_i to R_{i-1} are computed without access the RO. This happens when *congestion* causes link $(i, i+1)$ to go down for the duration of an exchange (so that R_i, who never calls the RO, has to compute all his upstream messages on his own).

Our next challenge is that Eve has no control, or even knowledge, of when congestion causes this event to occur. Indeed, the distribution generated by A', B' is only guaranteed to be close to the honest transcript overall; there is no guarantee that it is close to the honest transcript *conditioned on congestion on* $(i, i+1)$. Fortunately, we can show that with probability ρ^r, A', B' will generate a "useful" impersonated transcript that is ε/ρ^r-statistically close to the honest transcripts conditioned on the event that link $(i, i+1)$ is down. Eve does not necessarily know *when* she impersonates a useful transcript; she simply has to hope that she is lucky enough for this to happen.

The last challenge is that even when Eve is lucky enough to obtain a useful transcript, we still need a guarantee that (a) conditioned on B' generating a useful transcript, using B' to *interact* with the honest algorithm R_{i-1} results in a transcript that is statistically close to (b) the transcript between honest algorithms R_{i-1} and R_i conditioned on link $(i, i+1)$ being down. Unfortunately, the Naor-Rothblum algorithm does not give any guarantees when an honest algorithm *interacts* with an impersonated algorithm for more than 1 round. Thus, we prove that, with probability at least $(\rho/2)^r$, the impersonator algorithm B' interacting with honest $\mathsf{Alice}, ...R_{i-1}$ still generates a useful transcript such that the statistical distance between (a) and (b) is at most 1/100. (This assumes we take ε small enough; $\varepsilon = (\rho/10)^{4r} = 1/\text{poly}(n)$ suffices.)

To summarize, with probability $\geq 99/100$ Eve obtains algorithms A', B' from the Naor-Rothblum algorithm that can successfully impersonate *all* the honest algorithms. Then, with probability roughly $(\rho/2)^r$, she can use B' to *interact* with R_{i-1} as in Step 2 to drop a packet at R_{i-1} and generate a *useful* impersonated transcript that is 1/100-statistically close to the honest transcript produced when R_{i-1} and R_i interact conditioned on link $(i, i+1)$ being down. This breaks security with non-negligible probability, since link $(i, i+1)$ is not adjacent to Eve at R_{i-1}. ■

Statistical security. Our negative results in the statistical setting are more subtle. First of all, from [9, 8] the analog of Theorem 4 also holds, showing that the *entire system* needs to "perform cryptography". However, we run into trouble when we try to show that cryptography is required at *each intermediate node*. It turns out that Definition 3 does *not* inherently require complexity-based cryptography at intermediate nodes. We sketch a statistically secure FL protocol where the intermediate nodes $R_1, \ldots, R_K$ use only information-theoretically secure primitives (although Alice and Bob still use regular MAC's). While this protocol is completely impractical in terms of communication and storage overhead, we present it here to demonstrate the subtleties of Definition 3.[8]

Remark 1 (Impractical "crypto-free" statistical FL protocol.). The protocol uses one-time MACs (OTMAC), information-theoretic objects that have the same properties as regular MACs except that they can only be used a single time. (OTMACs and can be constructed from Carter-Wegman hashing.) Each node R_i shares pairwise keys with Alice. All the intermediate nodes and Bob store each packet that Alice sends to Bob. For each packet, Bob replies with an ack signed using a regular MAC. At the end of the interval, Alice counts the number of acks that she either fails to receive, or are invalid. The first time this count exceeds a β-fraction, Alice sends a "report request" message that is signed using a OTMAC to $R_1, \ldots, R_K, R_{K+1}$. Each node $R_1, \ldots, R_K$ responds with a report of every single packet they have witnessed, that is "onion signed" using the OTMAC (as in Section 3.1). Alice uses these reports in the usual way to localize link ℓ adjacent to Eve. From this point onwards Alice simply counts valid acknowledgments from Bob, and blames link ℓ each time the count exceeds a β fraction.

[8] In concurrent work, Wong et al. [24] propose a statistical FL scheme where no cryptography is performed *during an interval*. Instead, they precompute shared secrets that are appended to packets over the course of an interval and are used guarantee security. The secrets must refreshed periodically, which requires cryptographic participation by the intermediate nodes. This contrasts with the impractical scheme we describe here, which truly *never* requires any intermediate node to perform crypto.

The protocol satisfies Definition 3 because the probability that the failure rate at any link exceeds β by congestion alone is negligible. Since we do not allow Eve to move during the security game, if Alice successfully localizes Eve to link ℓ once, it means it must have been Eve's fault, and so from then on Alice can always blame all failures on link ℓ. As noted above, similar "impractical" protocols exist for per-packet protocols with $\omega(\log n)$ additional messages per packet (since all $\omega(\log n)$ messages are lost to congestion with only negligible probability), except that we replace the idea of "exceeding β fraction of failures" with "losing an entire exchange due to congestion". We may interpret this as follows:

1. It is unreasonable to assume that the failure rate at a link exceeds β only due to adversarial behaviour (*i.e.* Eve). For example, occasionally congestion might spike, or a router might malfunction or go down due maintenance, causing more than a β-fraction of packets to be dropped. If we assume such events happen with non-negligible probability, we can adapt the proof of Theorem 4 to show that cryptography is necessary at intermediate nodes for statistical security. As a corollary, if Eve can control congestion at links she does not occupy, then we need cryptography at every intermediate node. Our FL protocols remain secure even under the strongest such definition, where the failure rate on a link not occupied by Eve can exceed β.
2. We can take this issue outside of our model. If we say that it is reasonable that Eve cannot move during the security game, and that the failure rate cannot exceed β on a link that Eve does not control, then, as we showed above, there exist protocols where the intermediate nodes do not use complexity-based cryptography. However, we must be cognizant that in the real world there can be multiple adversaries that we would like to localize correctly, or the adversary may be able to move from one link to another. If protocols that do not use cryptography at intermediate nodes are to remain secure after Eve moves (and learns the key of previous nodes she occupied), then the keys at each node should be refreshed periodically. This key refresh process would require each intermediate node to use cryptography.

5 Open Problems

We gave lower bounds on the key-management and cryptographic overhead of secure FL protocols. The problem of bounding the *storage* requirements in an FL protocol is also still open. Furthermore, our results here only apply to FL on *single symmetric paths* between a single sender-receiver pair. An interesting question would be to consider FL for *asymmetric paths*, where the packets Bob sends back to Alice may take a different path than the packets that Alice sends to Bob. Another interesting direction is to consider FL in networks where packets can travel simultaneously on *multiple paths*, as in the SMT framework [6].

Acknowledgements. We thank Jennifer Rexford and Eran Tromer for very useful discussions and collaborations on failure localization and routing security in general. Shai Halevi and the anonymous EuroCrypt reviewers made a number of very helpful comments to improve the presentation of this paper. Boaz Barak is supported by NSF grants CNS-0627526 and CCF-0426582, US-Israel BSF grant 2004288 and Packard and Sloan fellowships. Sharon Goldberg is supported by NSF grant CNS-0627526. David Xiao is supported by a NSF Graduate Research Fellowship and a NDSEG Graduate Fellowship, and part of this work was done while visiting the Institute for Pure and Applied Mathematics at UCLA.

References

1. Traceroute (September 2001), http://costard.lbl.gov/cgi-bin/man/man2html?traceroute+8
2. Argyraki, K., Maniatis, P., Cheriton, D., Shenker, S.: Providing packet obituaries. ACM HotNets-III (2004)
3. Avramopoulos, I., Kobayashi, H., Wang, R., Krishnamurthy, A.: Highly secure and efficient routing. In: INFOCOM 2004, vol. 1, p. 208 (2004)
4. Awerbuch, B., Holmer, D., Nita-Rotaru, C., Rubens, H.: An secure routing protocol resilient to byzantine failures. In: WiSE 2002, pp. 21–30. ACM Press, New York (2002)
5. Crovella, M., Krishnamurthy, B.: Internet Measurement. Wiley, Chichester (2006)
6. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. J. ACM 40(1), 17–47 (1993)
7. Duffield, N.G., Grossglauser, M.: Trajectory sampling for direct traffic observation. IEEE/ACM Trans. Netw. 9(3), 280–292 (2001)
8. Goldberg, S., Xiao, D., Barak, B., Rexford, J.: A cryptographic study of secure internet measurement, TR-783-07. Technical report, Princeton University, Department of Computer Science (May 2007)
9. Goldberg, S., Xiao, D., Tromer, E., Barak, B., Rexford, J.: Path-quality monitoring in the presence of adversaries. In: ACM SIGMETRICS (2008)
10. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
11. He, J., Rexford, J.: Towards Internet-wide multipath routing. IEEE Network Magazine Special Issue on Scalablity (March 2008)
12. IETF. Network time protocol (ntp) charter, http://www3.ietf.org/proceedings/05mar/ntp.html
13. Impagliazzo, M., Luby, R.: One-way functions are essential for complexity based cryptography. In: FOCS, pp. 230–235 (October 1989)
14. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC 1989, pp. 44–61. ACM Press, New York (1989)
15. Jacobson, V.: Congestion avoidance and control. In: ACM SIGCOMM 1988, Stanford, CA, pp. 314–329 (Augest 1988)
16. Kent, S., Lynn, C., Seo, K.: Secure border gateway protocol (S-BGP). IEEE Jour. Sel. Areas in Comm. 18(4), 582–592 (2000)
17. Laskowski, P., Chuang, J.: Network monitors and contracting systems: competition and innovation. In: SIGCOMM 2006, pp. 183–194. ACM Press, New York (2006)
18. Mahajan, R., Spring, N., Wetherall, D., Anderson, T.: User-level internet path diagnosis. SIGOPS Oper. Syst. Rev. 37(5), 106–119 (2003)

19. Mizrak, A., Cheng, Y.-C., Marzullo, K., Savage, S.: Fatih: detecting and isolating malicious routers. In: DSN 2005, pp. 538–547 (28 June-1 July 2005)
20. Naor, M., Rothblum, G.N.: Learning to impersonate. In: ICML 2006, pp. 649–656. ACM Press, New York (2006)
21. Padmanabhan, V.N., Simon, D.R.: Secure traceroute to detect faulty or malicious routing. SIGCOMM Comput. Commun. Rev. 33(1), 77–82 (2003)
22. HÅstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
23. Subramanian, L., Roth, V., Stoica, I., Shenker, S., Katz, R.H.: Listen and whisper: security for BGP. In: USENIX NSDI 2004, p. 10 (2004)
24. Wong, E.L., Balasubramanian, P., Alvisi, L., Gouda, M.G., Shmatikov, V.: Truth in advertising: Lightweight verification of route integrity. In: PODC (2007)

A Vulnerabilities of Other FL Protocols

We sketch why the protocols of [21,4,2] do not satisfy our security definition.

An On-demand Secure Routing Protocol Resilient to Byzantine Failures [4]: Awerbuch, Holmer, Nita-Rotaru and Rubens present a statistical FL protocol in which Alice and Bob run a secure *failure detection* protocol, where Bob sends out authenticated acks for each packet he receives. Once the number of Packet delivery failures exceeds some threshold, say β, then Alice appends a encrypted list of "probed nodes" to each *new* packet that she sends out. If a node is included in the list of probed nodes, it is expected to send Alice an ack when it receives the packet containing the list. The acks are formed as our "onion reports". To localize failures, Alice chooses probed nodes according to a binary search algorithm, until she localizes a single link.

Now, consider an adversary Eve that sits at R_i and, for every sent packet where R_i is not included in the list of probed nodes, Eve happily causes failures. Eve stops causing failures whenever R_i is included in the list of probed nodes. Alice will never be able to localize such an Eve to a single link; as long as Eve behaves herself when she is part of the list of probed nodes, Alice has no way to find her. Our protocols avoid this problem by running their "detection phases" and "localization phases" on the same set of packets.

Furthermore, care must be taken in implementing this protocol in the presence of both adversarial behaviour and benign congestion. To see why, suppose that Eve causes the protocol to enter the localization phase. In [4], the binary search algorithm proceeds by one step each time failures are detected. It is important to ensure that normal congestion (on a link that is not adjacent to Eve) cannot cause the binary search algorithm to search for Eve in the wrong part of the path. To do this, the binary search algorithm should proceed by one step only when the *failure rate* exceeds some carefully chosen false alarm threshold (related to loss rate caused by normal congestion and the length of the portion of path that is currently being searched).

Packet Obituaries [2]: Argyraki, Maniatis, Cheriton, and Shenker propose an FL protocol that is similar to our Optimistic Protocol of Section 3.1. Each node locally stores digests of the packets they see, and at the end of some time interval, nodes send out reports to Alice that contain these packet digests. Alice then uses the information from these reports to localize failures on the path. The designers of this protocol focused on the benign setting, but mentioned that reports should also be *individually authenticated*. However, because these reports are not formed in a onion manner (as in our Optimistic Protocol) an adversarial node can implicate a innocent downstream node by selectively dropping the innocent node's reports.

Secure Traceroute [21]: We sketch attacks in the full version; this protocol has many of the same problems as [4,2].

$HB^{\#}$: Increasing the Security and Efficiency of HB^{+}

Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin

Orange Labs, 38–40 rue General Leclerc, Issy les Moulineaux, France
{henri.gilbert,matt.robshaw,yannick.seurin}@orange-ftgroup.com

Abstract. The innovative HB^{+} protocol of Juels and Weis [10] extends device authentication to low-cost RFID tags. However, despite the very simple on-tag computation there remain some practical problems with HB^{+} and despite an elegant proof of security against some limited active attacks, there is a simple man-in-the-middle attack due to Gilbert *et al.* [8]. In this paper we consider improvements to HB^{+} in terms of both security and practicality. We introduce a new protocol that we denote RANDOM-$HB^{\#}$. This proposal avoids many practical drawbacks of HB^{+}, remains provably resistant to attacks in the model of Juels and Weis, and at the same time is provably resistant to a broader class of active attacks that includes the attack of [8]. We then describe an enhanced variant called $HB^{\#}$ which offers practical advantages over HB^{+}.

Keywords: HB^{+}, RFID tags, authentication, LPN, Toeplitz matrix.

1 Introduction

The deployment of low-cost RFID tags is gathering pace. One familiar application is the inventory tracking of consumer items such as clothes, media products, and pharmaceuticals. However since blank tags can be programmed, there are opportunities for an attacker to clone an RFID tag and to introduce counterfeit goods into the supply chain. Thus, in this and other application areas there is much interest in deploying mechanisms for cryptographic tag authentication. However the physical demands for the deployment of cryptography on a cheap tag are substantial. Not only is space limited [10], but the peak and average power consumption often pose a demanding barrier for a tag that derives its power from a reader. Furthermore, since RFID tags pass fleetingly past a reader and are used in multi-tag and multi-reader environments, the communication is limited and its coordination complex.

Juels and Weis introduced HB^{+}, a three-pass symmetric key authentication protocol, at Crypto 2005 [10]. HB^{+} is computationally lightweight—requiring only simple bit-wise operations—and it is supported by a proof of security [10]. There are, however, some practical deficiencies in HB^{+} and the value of the proof of security has been somewhat limited by a simple active attack due to Gilbert *et al.* [8] which we will refer to as the GRS attack. Nevertheless, the simplicity of both the original proposal and the active attack have led to a number of HB-related publications (see Section 2.2).

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 361–378, 2008.

Tag (secret $\boldsymbol{x}, \boldsymbol{y}$)		Reader (secret $\boldsymbol{x}, \boldsymbol{y}$)
$\nu \in_R \{0, 1 \vert \text{Prob}(\nu = 1) = \eta\}$		
Choose $\boldsymbol{b} \in_R \{0,1\}^k$	$\xrightarrow{\boldsymbol{b}}$	
	$\xleftarrow{\boldsymbol{a}}$	Choose $\boldsymbol{a} \in_R \{0,1\}^k$
Let $z = \boldsymbol{a} \cdot \boldsymbol{x}^t \oplus \boldsymbol{b} \cdot \boldsymbol{y}^t \oplus \nu$	$\xrightarrow{z}$	Check $\boldsymbol{a} \cdot \boldsymbol{x}^t \oplus \boldsymbol{b} \cdot \boldsymbol{y}^t = z$

Fig. 1. One single round of HB^+ [10]. The entire authentication process requires r rounds and, in this basic form, each round consists of the three passes shown. Provided the tag fails less than some threshold t number of rounds, the tag is authenticated.

In this paper we propose solutions that improve on the practical problems of HB^+ while providing resistance to the GRS attack. The two simple proposals RANDOM-$\text{HB}^\#$ and $\text{HB}^\#$ provide more practical error rates than the original HB^+ and reduce the communication payload by a factor of around 20 (depending on the parameter sets). The protocol RANDOM-$\text{HB}^\#$ is provably secure in the *detection-based* model, the adversarial model used in *all* current proofs of security for HB^+ and its variants. But RANDOM-$\text{HB}^\#$ is also provably secure against the GRS attack and more generally in what we term the GRS-MIM model, an adversarial model that permits an active adversary to manipulate messages from the reader. The related protocol $\text{HB}^\#$ then gives a truly efficient scheme. While the same proofs do not immediately extend in their entirety to $\text{HB}^\#$, we can still say a surprising amount about the scheme in both theory and practice.

Our paper is organised as follows. First we describe HB^+ and some variants. Then, in Section 3, we introduce RANDOM-$\text{HB}^\#$ and provide full security proofs. In Section 4 we describe $\text{HB}^\#$ and its security and practical performance. We then highlight future work and draw our conclusions. Throughout we aim to use established notation. There will be some interplay between vectors $\boldsymbol{x} \in \{0,1\}^k$ (which we always consider to be row vectors) and scalars in GF(2). We use bold type $\boldsymbol{x}$ to indicate a row vector while scalars x are written in normal text. The bitwise addition of two vectors will be denoted $\oplus$ just as for scalars. We denote the *Hamming weight* of $\boldsymbol{x}$ by $\text{Hwt}(\boldsymbol{x})$.

2 HB^+ Variants and Tag Authentication

There are now several protocols based on HB^+ and these offer a variable level of security and practicality. We start by reviewing the original protocol. HB^+ is a three-pass authentication protocol built on the conjectured hardness of the *Learning from Parity with Noise* (LPN) problem [10].

> **LPN Problem.** Let A be a random $(q \times k)$-binary matrix, let $\boldsymbol{x}$ be a random k-bit vector, let $\eta \in]0, \frac{1}{2}[$ be a noise parameter, and let be a random q-bit vector such that $\text{Hwt}(\;) \leq \eta q$. Given A, η, and $\boldsymbol{z} = A \cdot \boldsymbol{x}^t \oplus {}^t$, find a k-bit vector $\boldsymbol{y}^t$ such that $\text{Hwt}(A \cdot \boldsymbol{y}^t \oplus \boldsymbol{z}) \leq \eta q$.

The HB$^+$ protocol is outlined in Figure 1. One doesn't need to look long to see that the goal of low on-tag computation has been achieved. Leaving aside generating $\boldsymbol{b}$ and the bit ν, computation on the tag is reduced to a dot-product (which can be computed bit-wise) and a single bit exclusive-or. Also HB$^+$ is accompanied by a proof of security. The adversarial model for this proof is referred to as the *detection-based* model [10] and requires that the adversary queries a tag q times and then attempts to pass the HB$^+$ authentication process by interacting with the reader once. Some commentators are not convinced that this adversarial model is sufficiently strong and an active attack against HB$^+$ exists when the adversary can interact with both the tag and the reader before attempting to impersonate the tag [8]. That said, the proof of security still has considerable value. The original proof [10] was rather sophisticated and applied to an adversary attempting to fool the reader over a single round of HB$^+$. This was extended by Katz and Shin [12] who also considered the parallel version of HB$^+$ with communications batched into one round of a three-pass protocol.

2.1 Some Problems with HB$^+$

While HB$^+$ is computationally lightweight it still has some practical defects. The possibility of a legitimate tag being rejected has been commented on [12], but other issues such as the complex and extensive tag-reader communication would make HB$^+$ difficult to use. First, however, we highlight the fact that methods to solve the LPN problem have improved since the original presentation of HB$^+$.

LPN security and parameter choices. When considering the security and implementation of HB$^+$ there are four parameters that we need to set:

k :	the length of the secrets,	η :	the noise level,
r :	the number of rounds,	t :	the threshold for tag acceptance.

The first two parameters, k and η, quantify the resistance of the underlying LPN problem to attack. In [11] it is suggested that the parameter sets $k = 224$ and $\eta = 0.25$ provide around 80-bit security. Katz and Shin [12] propose $k \approx 200$ with $\eta = 0.125$, but we note that the reduced level of noise means that the LPN problem instance becomes easier and would necessitate an increase[1] to k.

Since the publication of HB$^+$ the LPN problem has been studied in more detail and the BKW algorithm cited in [10,12] has been improved. Fossorier *et al.* [6] show that the parameter choices used by [10] offer a level of security no greater than 2^{61} operations rather than the 2^{80} claimed. However, this has been superseded by the work of Levieil and Fouque [16] which suggests that the real security level offered by the parameters in [10] is no more than 2^{52} operations. Considering [16] we propose alternative parameter values in Section 4.2 that are more consistent with the intended security level. In particular we propose $k = 512$ and $\eta = 0.125$ or, more conservatively, $k = 512$ and $\eta = 0.25$.

[1] However [12] is concerned with security proofs and specific parameter choices are somewhat orthogonal to their work.

Table 1. Error rates and transmission costs for different parameter sets in HB$^+$. The threshold $t = r\eta$ is proposed in [10] so we use $\lceil r\eta \rceil$ in this table. For the other parameters, [10] suggest $k = 224$ and $\eta = 0.25$ (leaving r unspecified) while [12] suggests $k \approx 200$, $\eta = 0.125$, with $40 \leq r \leq 50$. Based on the work of [16], we also consider the data transmission costs when $k = 512$ in the last column.

r	η	k	False reject rate (P_{FR})	False accept rate (P_{FA})	Transmission cost (bits) [k as given]	[$k = 512$]
80	0.25	224	0.44	4×10^{-6}	35,920	82,000
60	0.25	224	0.43	6×10^{-5}	26,984	61,500
40	0.25	224	0.42	1×10^{-3}	17,960	41,000
50	0.125	200	0.44	2×10^{-8}	20,050	51,250
40	0.125	200	0.38	7×10^{-9}	16,040	41,000

Error rates. A false rejection, a legitimate tag being rejected by a legitimate reader, occurs when the number of incorrect authentications exceeds the threshold t. A false acceptance takes place when an illegitimate tag is accepted by a legitimate reader. This occurs when t or fewer verification errors take place and we assume the illegitimate tag is reduced to guessing the reply z at random. The probability of a false rejection, P_{FR}, and a false acceptance, P_{FA}, are given by

$$P_{\mathrm{FR}} = \sum_{i=t+1}^{r} \binom{r}{i} \eta^i (1-\eta)^{r-i} \text{ and } P_{\mathrm{FA}} = \sum_{i=0}^{t} \binom{r}{i} 2^{-r}.$$

Note that both the false rejection and acceptance rate are independent of k, the size of the secrets, while the false acceptance rate is also independent of the noise level η used in HB$^+$. In the original descriptions of HB$^+$ a threshold of $t = r\eta$ is suggested. However (see Table 1) such a choice gives an unacceptably high false rejection rate. It is hard to imagine any practical scenario where a probability higher than 1% of rejecting a legitimate tag could be tolerated.

Transmission costs. HB$^+$ is a three-pass protocol that runs over r rounds. This requires the exchange of $2k+1$ bits per round and $2rk+r$ bits in total. In the parallel version of the protocol, the data transmission requirements are the same but the data is packed into three passes of rk, rk, and r bits respectively. A three-pass protocol is considerably more practical than a $3r$-pass protocol (this was also mentioned in [12] as a justification for parallel HB$^+$). However the total amount of data transferred in both cases remains unacceptably high. In Table 1 we provide some estimates for the transmission costs in using HB$^+$. In particular we use parameter values that cover those proposed in [10,12]. We also include the transmission costs if we were to use parameter sizes that come closer to providing the intended 80-bit level of security.

An active attack. A simple active attack on HB$^+$ was provided in [8]. There it is assumed that an adversary can manipulate challenges sent by a legitimate reader to a legitimate tag during the authentication exchange, and can learn

Tag (secret $\boldsymbol{x}, \boldsymbol{y}$)		Reader (secret $\boldsymbol{x}, \boldsymbol{y}$)
$\nu \in \{0, 1 \vert \text{Prob}(\nu = 1) = \eta\}$		
Choose $\boldsymbol{b} \in_R \{0,1\}^k$	$\xrightarrow{\boldsymbol{b}}$	
	$\xleftarrow{\boldsymbol{a'} = \boldsymbol{a} \oplus} \cdots \xleftarrow{\boldsymbol{a}}$	Choose $\boldsymbol{a} \in_R \{0,1\}^k$
Let $z' = \boldsymbol{a'} \cdot \boldsymbol{x}^t \oplus \boldsymbol{b} \cdot \boldsymbol{y}^t \oplus \nu$	$\xrightarrow{z'}$	Check $\boldsymbol{a} \cdot \boldsymbol{x}^t \oplus \boldsymbol{b} \cdot \boldsymbol{y}^t = z'$

Fig. 2. The attack of Gilbert *et al.* [8] on HB$^+$. The adversary modifies the communications between reader and tag (by adding some perturbation) and notes whether authentication is still successful. This reveals one bit of secret information.

whether such manipulation gives an authentication failure. The attack consists of choosing a constant k-bit vector and using it to perturb the challenges sent by a legitimate reader to the tag; is exclusive-or'ed to each authentication challenge for each of the r rounds of authentication. If the authentication process is successful then we must have that $\cdot \boldsymbol{x}^t = 0$ with overwhelming probability. Otherwise $\cdot \boldsymbol{x}^t = 1$ with overwhelming probability and acceptance or rejection by the reader reveals one bit of secret information. The attack is illustrated in Figure 2 for one round of the HB$^+$ protocol. To retrieve the k-bit secret $\boldsymbol{x}$, one can repeat the attack k times for linearly independent 's and solve the resulting system. Conveniently, an adversary can choose 's with a single non-zero bit. With $\boldsymbol{x}$ an attacker can impersonate the tag by setting $\boldsymbol{b} = \boldsymbol{0}$. Alternatively, an attacker can emulate a false tag using $\boldsymbol{x}$, send a chosen blinding factor $\boldsymbol{b}$ to a legitimate reader, and return $\boldsymbol{a} \cdot \boldsymbol{x}^t$ to the challenge $\boldsymbol{a}$. If authentication is successful $\boldsymbol{b} \cdot \boldsymbol{y}^t = 0$, otherwise $\boldsymbol{b} \cdot \boldsymbol{y}^t = 1$, with overwhelming probability, and $\boldsymbol{y}$ can be recovered with k linearly independent $\boldsymbol{b}$.

Whether or not the attack is technically easy to mount it is *certificational*. The attack is mathematically simple and fully compromises HB$^+$. Protocols that resist this attack, while maintaining the computational simplicity of HB$^+$, would therefore be very attractive.

2.2 Other Work on HB$^+$ and Tag Authentication

The novelty of the HB$^+$ protocol has generated considerable interest and much research. We have already mentioned the work of Katz and Shin [12] that closed gaps and extended the original proof of security. Follow-on work by Katz and Smith [13] has further extended these theoretical results to a larger range of noise levels $\frac{1}{4} \leq \eta < \frac{1}{2}$ whereas previous work [12] was only valid for $\eta < \frac{1}{4}$.

Other researchers have considered the active attack of Gilbert *et al.* [8]. Among them Bringer *et al.* [2] have outlined a protocol named HB^{++}. However the resulting protocol has some practical drawbacks. The data transmission costs of HB$^+$ remain and the on-tag computation now includes bit-wise rotations and a small-block permutation f. Furthermore, an additional pre-protocol involving a universal hash function h is required to derive new tag/reader secrets at the start of each authentication. All this requires additional hardware and moves

Parameters: (k_X, k_Y, m, η, u)		
Tag (secret X, Y)		Reader (secret X, Y)
$\in_R \{\{0,1\}^m \mid$ Prob.$(\nu_i = 1) = \eta$ for $1 \leq i \leq m\}$		
Choose $\boldsymbol{b} \in_R \{0,1\}^{k_Y}$	$\xrightarrow{\boldsymbol{b}}$	
	$\xleftarrow{\boldsymbol{a}}$	Choose $\boldsymbol{a} \in_R \{0,1\}^{k_X}$
Let $\boldsymbol{z} = \boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y \oplus$	$\xrightarrow{\boldsymbol{z}}$	Check $\text{Hwt}(\boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y \oplus \boldsymbol{z}) \leq um$

Fig. 3. The RANDOM-HB$^\#$ authentication protocol where the secrets X and Y are binary random matrices and the protocol has a single round. The verification step requires the comparison of two vectors and yields a PASS/FAIL verdict.

away from the essential simplicity of the HB$^+$ protocol. Piramuthu [20] proposes a modification to HB^{++} in which the bit-wise rotations are varied for each round of the authentication and the message flow is simplified (saving one bit per authentication round). However the exact security claims are unclear. The variant HB* is proposed by Duc and Kim [4] while another prominent protocol is HB-MP [19]. While both claim to be resistant to the attacks of [8], linear time attacks by the authors [7] show that this is not the case.

Naturally, research into other mechanisms for unilateral and mutual authentication continue in parallel. Schemes based on symmetric cryptography might use a lightweight block cipher [1,21] in a challenge-response protocol while other schemes might use asymmetric techniques such as GPS [9,18]. Other proposals include SQUASH [22] which might be viewed as a dedicated MAC, though the security goals appear to be somewhat reduced when compared to HB$^+$ and the proposals RANDOM-HB$^\#$ and HB$^\#$ in this paper.

But this parallel work only serves to emphasize the interest in tag authentication and the importance of understanding the limits of proposals like HB$^+$. Despite the mixed success of current proposals in the literature, HB$^+$ still holds much promise. This is due to the exceptionally low on-tag computational requirements and the fact that a proof of security, even if the model is weaker than we might ideally like, is a positive attribute.

3 The Proposal RANDOM-HB$^\#$

We now introduce RANDOM-HB$^\#$ (RANDOM-*HB-sharp*). This goes a long way to fixing many of the practical problems of HB$^+$. Like many other HB$^+$-variants, we prove the security of RANDOM-HB$^\#$ in the *detection-based* model, referred to in what follows as the DET-model. But we go further and prove the security of RANDOM-HB$^\#$ against a class of attacks that includes the GRS attack in what we term the GRS-MIM-model. More details are given in Section 3.1, but this model allows an active attacker to change any message from the reader in any way that they wish and observe the decision of the reader of whether to accept or not.

In RANDOM-HB$^{\#}$ we generalise HB^{+} and change the form of the secrets $\boldsymbol{x}$ and $\boldsymbol{y}$ from k-bit vectors into $(k_X \times m)$- and $(k_Y \times m)$-binary matrices X and Y. We illustrate RANDOM-HB$^{\#}$ protocol in Figure 3. One way of looking at RANDOM-HB$^{\#}$ is to observe that it is equivalent to m iterations of HB^{+}, but each column of X and Y in RANDOM-HB$^{\#}$ effectively represents a different HB^{+} secret $\boldsymbol{x}$ and $\boldsymbol{y}$. However, while RANDOM-HB$^{\#}$ carries much of the appearance of the HB^{+} protocol, there are important differences. In particular, the final verification by the reader consists of the comparison of two m-bit vectors $\boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y$ and $\boldsymbol{z}$. For reader-verification we merely count the number of positions e that are in error and if $e \leq t$ for some threshold $t = um$, where $u \in]\eta, \frac{1}{2}[$, then we deduce that the tag is authentic. Thus RANDOM-HB$^{\#}$ and HB$^{\#}$ (see Section 4) consist of a single round.

3.1 Security Results for RANDOM-HB$^{\#}$

We now provide security proofs for RANDOM-HB$^{\#}$ in two models. The first is the DET-model used in much of the founding work on HB^{+} [10,12]. Here the adversary is only allowed to query an honest tag without access to the reader. The second permits an active attacker to manipulate messages sent by the reader and will be referred to as the GRS-MIM-model.

Security definitions. In the following, the security parameter will be k, to which the number of rows of the secret matrices X and Y are related by $k_X = \Theta(k)$ and $k_Y = \Theta(k)$. We will say that a function (from positive integers to positive real numbers) is *negligible* if it approaches zero faster than any inverse polynomial, and *noticeable* if it is larger than some inverse polynomial. An algorithm will be *efficient* if it is a *Probabilistic Polynomial-Time* Turing machine. By saying that LPN is a hard problem, we mean that any efficient adversary solves it with only negligible probability.

We will let $\mathcal{T}_{X,Y,\eta}$ denote the algorithm run by an honest tag in the RANDOM-HB$^{\#}$ protocol and $\mathcal{R}_{X,Y,u}$ the algorithm run by the tag reader. We will prove the security of RANDOM-HB$^{\#}$ in two models:

- The DET-model, defined in [10,12], where attacks are carried out in two phases: the adversary first interacts q times with the honest tag. Then the adversary interacts with the reader and tries to impersonate the valid tag.
- The GRS-MIM-model: in a first phase, the adversary can eavesdrop on all communications between an honest tag and an honest reader (including the reader-decision of whether to accept or not) and in addition the attacker can modify any message from the reader to the tag for q executions of the protocol. Then the adversary interacts only with the reader and tries to impersonate the valid tag.

Note that the DET-model is a restriction of the GRS-MIM-model as any attack in the DET-model can easily be converted into an attack in the GRS-MIM-model. By replying at random to a challenge, the probability an adversary impersonating a tag will succeed is the false acceptance rate $P_{\text{FA}} = 2^{-m} \sum_{i=0}^{um} \binom{m}{i}$. This quantity is the best soundness we can achieve for RANDOM-HB$^{\#}$. Note that it is a function of m and u and not of the security parameter k, which will only set how close to P_{FA} the advantage of an adversary is bound to be. Note also that P_{FA} is negligible for any $u \in]\eta, \frac{1}{2}[$ and any $m = \Theta(k)$. We define the advantage of an adversary against the RANDOM-HB$^{\#}$ protocol in the DET and GRS-MIM models as its overhead success probability over P_{FA} in impersonating the tag:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{DET}}(k_X, k_Y, m, \eta, u, q) \stackrel{\mathrm{def}}{=}$$
$$\Pr\left[X \stackrel{\$}{\leftarrow} \mathcal{M}_X, Y \stackrel{\$}{\leftarrow} \mathcal{M}_Y,\ \mathcal{A}^{\mathcal{T}_{X,Y,\eta}}(1^k) \,:\, \langle \mathcal{A}, \mathcal{R}_{X,Y,u} \rangle = \mathrm{ACC}\right] - P_{\mathrm{FA}};$$
$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{GRS\text{-}MIM}}(k_X, k_Y, m, \eta, u, q) \stackrel{\mathrm{def}}{=}$$
$$\Pr\left[X \stackrel{\$}{\leftarrow} \mathcal{M}_X, Y \stackrel{\$}{\leftarrow} \mathcal{M}_Y,\ \mathcal{A}^{\mathcal{T}_{X,Y,\eta}, \mathcal{R}_{X,Y,u}}(1^k) \,:\, \langle \mathcal{A}, \mathcal{R}_{X,Y,u} \rangle = \mathrm{ACC}\right] - P_{\mathrm{FA}}.$$

where $\mathcal{M}_X$ and $\mathcal{M}_Y$ denote resp. the sets of $(k_X \times m)$- and $(k_Y \times m)$-binary matrices and ACC denotes "accept".

Proof methods. We do not reduce the security of RANDOM-HB$^{\#}$ directly to the LPN problem. A preliminary step of our analysis is to define a natural matrix-based extension of the LPN problem and to prove its hardness. For this we appeal to the theory of "weakly verifiable puzzles". This is a notion introduced by Canetti, Halevi, and Steiner [3] and, informally, refers to a situation where only the entity that generates the puzzle holds secret information enabling the correctness of a candidate solution to be efficiently verified. As noticed by Katz and Shin [12], attacking the one-round HB protocol [10] in the passive model (that is, given q noisy samples $(\boldsymbol{a_i}, \boldsymbol{a_i} \cdot \boldsymbol{x}^t \oplus \nu_i)$, where $\boldsymbol{x}$ is a secret k-bit vector and the $\boldsymbol{a_i}$ are random k-bit vectors, and a random challenge $\boldsymbol{a}$, guess $\boldsymbol{a} \cdot \boldsymbol{x}^t$) may be viewed as a weakly verifiable puzzle. The result by Juels and Weis [10, Lemma 1] asserts, in essence, that this puzzle is $(1 - \frac{1}{2})$-hard if we assume the hardness of the LPN problem, which means that any efficient adversary trying to solve it has a success probability that is negligibly close (in k) to $\frac{1}{2}$. Canetti *et al.* [3] proved that if no efficient algorithm can solve a puzzle with probability more than ϵ, then no efficient algorithm can solve m independent puzzles simultaneously with probability more than ϵ^m. Thus, we define an extension of the HB puzzle that we call the *MHB puzzle*: given q noisy samples $(\boldsymbol{a_i}, \boldsymbol{a_i} \cdot X \oplus \ {}_{\boldsymbol{i}})$, where X is a secret $(k \times m)$-matrix and the $\boldsymbol{a_i}$ are random k-bit vectors, and a random challenge $\boldsymbol{a}$, guess $\boldsymbol{a} \cdot X$. Using Canetti *et al.*'s result, we prove that any efficient adversary trying to solve it has a success probability that is negligibly close (in k) to $\frac{1}{2^m}$. All the necessary definitions and results are given in the full version of this paper.[2]

The security analysis is carried out in two steps. First we reduce the security of RANDOM-HB$^{\#}$ in the DET-model to the MHB puzzle. Then we reduce the security in the GRS-MIM-model to the security in the DET-model.

[2] Available from http://eprint.iacr.org/2008/028

Theorem 1 (Security of RANDOM-HB$^{\#}$ in the DET-model). *Let $\mathcal{A}$ be an adversary attacking the* RANDOM-*HB$^{\#}$ protocol with parameters (k_X, k_Y, m, η, u) in the* DET-*model, interacting with the tag in at most q executions of the* RANDOM-*HB$^{\#}$ protocol, running in time T, and achieving advantage greater than δ. Then there is an adversary $\mathcal{A}'$, running in time at most $2mLq(2+\log_2 q)T$, solving the MHB puzzle with parameters (k_Y, m, η, q'), where $q' = mLq(2+\log_2 q)$ and $L = \frac{512}{\delta^4(1-2u)^4}(\ln m - \ln\ln 2)$, with success probability $> \left(\frac{1}{2^m} + \frac{\delta}{4}\right)$. Hence, assuming the hardness of the LPN problem, the advantage of any efficient* DET-*adversary against the* RANDOM-*HB$^{\#}$ protocol is negligible in k. As a consequence, for parameters $m = \Theta(k)$, the probability of any efficient* DET-*adversary to impersonate a valid tag is negligible in k.*

Proof. We slightly adapt the proof of Juels and Weis [11, Appendix C]. We denote by $\{(\boldsymbol{b_i}, \boldsymbol{z_i})\}_{1\le i\le q'}$ the set of samples obtained by $\mathcal{A}'$ from the MHB puzzle generator with secret matrix Y and $\boldsymbol{b}$ the challenge vector for which $\mathcal{A}'$ aims to output $\boldsymbol{z} = \boldsymbol{b}\cdot Y$. $\mathcal{A}'$ uses its samples to simulate a tag algorithm $\mathcal{T}_{X,Y,\eta}$ where X is random with one line equal to $\boldsymbol{z}$. $\mathcal{A}'$ proceeds as follows:

1. Choose a random j, $1 \le j \le k_X$, and construct the $k_X \times m$ matrix X' where all rows are random except the j-th one which is undefined (say, equal to zero). Let $\boldsymbol{x_l}$ denote the l-th row of X'.
2. Divide the $q' = mLq(1+r)$ samples $\{(\boldsymbol{b_i}, \boldsymbol{z_i})\}_{1\le i\le q'}$ into mL sets of $q(1+r)$ samples. For each bit position $s = 1$ to m, repeat the following L times, considering a fresh set of $q(1+r)$ samples each time:
   (a) For $i = 1$ to q repeat the following: draw a random bit α_i (this is a guess at the j-th bit of the challenge $\boldsymbol{a_i^+}$ which will be sent by the adversary $\mathcal{A}$). If $\alpha_i = 0$, send to $\mathcal{A}$ the blinding vector $\boldsymbol{b_i^+} = \boldsymbol{b_i}$, if $\alpha_i = 1$, send to $\mathcal{A}$ the blinding vector $\boldsymbol{b_i^+} = \boldsymbol{b_i} \oplus \boldsymbol{b}$. $\mathcal{A}$ sends back the challenge $\boldsymbol{a_i^+}$. If the guess was right (i.e. $\alpha_i = \boldsymbol{a_i^+}[j]$), then answer with the vector

   $$\boldsymbol{z_i^+} = \bigoplus_{l\ne j}\left(\boldsymbol{a_i^+}[l]\cdot \boldsymbol{x_l}\right) \oplus \boldsymbol{z_i}.$$

   Otherwise rewind adversary $\mathcal{A}$ to the beginning of its i-th query and try with a new $(\boldsymbol{b_{i'}}, \boldsymbol{z_{i'}})$ chosen among the rq supplementary samples.
   (b) If the rq samples are exhausted before the simulation of the query phase of $\mathcal{A}$ ends, randomly guess $\boldsymbol{z}[s]$.
   (c) Otherwise, go to the cloning phase of $\mathcal{A}$: $\mathcal{A}$ sends a blinding vector $\hat{\boldsymbol{b}}$. Choose two random challenge vectors $\hat{\boldsymbol{a}}_{\boldsymbol{1}}$ and $\hat{\boldsymbol{a}}_{\boldsymbol{2}}$ such that they differ in their j-th bit. Transmit $\hat{\boldsymbol{a}}_{\boldsymbol{1}}$ to $\mathcal{A}$, record its response $\hat{\boldsymbol{z}}_{\boldsymbol{1}}$, rewind the adversary, transmit $\hat{\boldsymbol{a}}_{\boldsymbol{2}}$ to $\mathcal{A}$, and record its response $\hat{\boldsymbol{z}}_{\boldsymbol{2}}$ as well.
   (d) Compute the guess for $\boldsymbol{z}[s]$ as

   $$\hat{\boldsymbol{z}}_{\boldsymbol{1}}[s] \oplus \hat{\boldsymbol{z}}_{\boldsymbol{2}}[s] \oplus \left(\bigoplus_{l\ne j}(\hat{\boldsymbol{a}}_{\boldsymbol{1}}[l] \oplus \hat{\boldsymbol{a}}_{\boldsymbol{2}}[l])\cdot \boldsymbol{x_l}[s]\right).$$

3. Once L guesses have been made for each m bits of $\boldsymbol{z}$, take the majority outcome for each of them and output the answer accordingly.

Let us analyse what $\mathcal{A}'$ achieves. The repeated experiments on $\mathcal{A}$ share some common randomness ω (namely X and Y). Let us denote by ω' the randomness "renewed" at each experiment (that is the randomness used to simulate the tag, the random challenge $\hat{\boldsymbol{a}}$, and $\mathcal{A}$'s internal randomness). By a standard averaging argument, it holds that with probability greater than $P_{\text{FA}} + \frac{\delta}{2}$ over ω, the answer returned by $\mathcal{A}$ is correct in at least $m - t$ positions with probability greater[3] than $\frac{\delta}{2}$ over ω'. Let us assume that this is the case and show that $\mathcal{A}'$ returns a correct answer $\boldsymbol{z}$ with probability greater than $\frac{1}{2}$. The theorem will follow since $P_{\text{FA}} > \frac{2}{2^m}$ as soon as $t > 1$ and the overall probability of success for $\mathcal{A}'$ will be greater than $\frac{P_{\text{FA}}}{2} + \frac{\delta}{4} > \frac{1}{2^m} + \frac{\delta}{4}$.

First we will show that, during phase 2(a), $\mathcal{A}'$ simulates a tag algorithm $\mathcal{T}_{X,Y,\eta}$, where X is the X' matrix with $\boldsymbol{z}$ as j-th row. To see this, observe that when $\alpha_i = \boldsymbol{a_i^+}[j] = 0$, then

$$\boldsymbol{z_i^+} = \boldsymbol{a_i^+} \cdot X \oplus \boldsymbol{b_i} \cdot Y \oplus \quad {}_{\boldsymbol{i}} = \boldsymbol{a_i^+} \cdot X \oplus \boldsymbol{b_i^+} \cdot Y \oplus \quad {}_{\boldsymbol{i}},$$

whereas when $\alpha_i = \boldsymbol{a_i^+}[j] = 1$, then

$$\boldsymbol{z_i^+} = \boldsymbol{a_i^+} \cdot X \oplus \boldsymbol{z} \oplus \boldsymbol{b_i} \cdot Y \oplus \quad {}_{\boldsymbol{i}} = \boldsymbol{a_i^+} \cdot X \oplus (\boldsymbol{b_i} \oplus \boldsymbol{b}) \cdot Y \oplus \quad {}_{\boldsymbol{i}} = \boldsymbol{a_i^+} \cdot X \oplus \boldsymbol{b_i^+} \cdot Y \oplus \quad {}_{\boldsymbol{i}}.$$

Let us now analyse the advantage $\mathcal{A}'$ enjoys during a single guess for one bit of $\boldsymbol{z}$ during phase 2. First, one can upper bound the probability that $\mathcal{A}'$ enters phase 2(b) by the probability that any one of the q experiments results in the discarding of r pairs of the extra challenge-response pairs, which is $q2^{-r}$. Taking $r = \log_2 q + 1$ yields a probability not greater than $1/2$.

Consider phase 2(d) for a fixed bit position s. The guess of $\mathcal{A}'$ is right when both bits $\hat{\boldsymbol{z}}_{\boldsymbol{1}}[s]$ and $\hat{\boldsymbol{z}}_{\boldsymbol{2}}[s]$ are correct, or when they are both incorrect. Hence we are interested in lower bounding the probability p' of this event. First, we will lower bound the probability p over ω' that the s-th bit of the answer returned by $\mathcal{A}$ is correct. We will assume *w.l.o.g.* that this probability is the same in all positions (otherwise one can "symmetrize" $\mathcal{A}$ by applying a random permutation of $\{1, \ldots, m\}$ to the problem). We can lower bound p as follows. Suppose we draw a *random* bit position s. Clearly, this bit is correct with probability p over the choice of s and ω'. At the same time, conditioned on the fact that more than $m - t$ bits are correct, the s-th bit of the answer is correct with probability greater than $1 - u$. Consequently, the overall probability for the s-th bit to be correct is greater than $(1-u)\frac{\delta}{2} + \frac{1}{2}(1 - \frac{\delta}{2})$, hence $p \geq \frac{1}{2} + \epsilon$ where $\epsilon = \frac{\delta}{2}(\frac{1}{2} - u)$. Juels and Weis proved [10, Lemma 2] that in this case, the probability, *conditioned on the fact that* $\hat{\boldsymbol{a}}_{\boldsymbol{1}}$ *and* $\hat{\boldsymbol{a}}_{\boldsymbol{2}}$ *differ in a single bit* j, that both bits $\hat{\boldsymbol{z}}_{\boldsymbol{1}}[s]$ and $\hat{\boldsymbol{z}}_{\boldsymbol{2}}[s]$ are correct or incorrect at the same time, is greater than $\frac{1}{2} + \epsilon^3/2 - (\epsilon^3 + 1)/k_X$. However one can improve on their analysis by using Jensen's inequality[4].

[3] Otherwise the probability of success of the adversary would be upper bounded by $(1 - P_{\text{FA}} - \frac{\delta}{2})\frac{\delta}{2} + P_{\text{FA}} + \frac{\delta}{2} < \delta + P_{\text{FA}}$, contradicting the hypothesis on $\mathcal{A}$.

[4] Note that this will also improve the security reduction for HB^+.

Let γ denote the randomness except for $\hat{\boldsymbol{a}}$ in the experiment ω' we are considering. For a fixed γ, let p_γ denote the probability over $\hat{\boldsymbol{a}}$ that the s-th bit of the answer from $\mathcal{A}$ is correct. We've just proved that $\sum_\gamma p_\gamma \geq \frac{1}{2} + \epsilon$. Let p'_γ denote for a fixed γ, the probability, *conditioned on the fact that* $\hat{\boldsymbol{a}}_{\mathbf{1}}$ *and* $\hat{\boldsymbol{a}}_{\mathbf{2}}$ *differ in a single bit* j, that both bits $\hat{\boldsymbol{z}}_{\mathbf{1}}[s]$ and $\hat{\boldsymbol{z}}_{\mathbf{2}}[s]$ are correct or incorrect at the same time. Following the proof of [10, Lemma 2] we have $p'_\gamma \geq \phi(p_\gamma)$ where

$$\phi(x) = x^2 \left(\frac{k_X + \log_2 x - 1}{k_X}\right) + (1-x)^2 \left(\frac{k_X + \log_2(1-x) - 1}{k_X}\right).$$

As ϕ is convex, one has the following inequalities:

$$p' = \sum_\gamma p'_\gamma \geq \sum_\gamma \phi(p_\gamma) \geq \phi(\sum_\gamma p_\gamma) = \phi(p) \geq \phi(\frac{1}{2} + \epsilon) \geq \frac{1}{2} + 2\epsilon^2 - \frac{1}{k_X}.$$

As $\mathcal{A}'$ enters phase 2(b) with probability less than $1/2$, the probability that $\mathcal{A}'$ guesses bit $\boldsymbol{z}[s]$ correctly is lower-bounded by $\frac{1}{4} + \frac{p'}{2} \geq \frac{1}{2} + \epsilon'$, with $\epsilon' = \epsilon^2 - \frac{1}{2k_X}$.

Using the Chernoff bound, taking the majority outcome of the L experiments allows $\mathcal{A}'$ to guess bit s with probability greater than

$$\pi = \left(1 - e^{\frac{-L\epsilon'^2}{1+2\epsilon'}}\right) \geq \left(1 - e^{\frac{-L\epsilon'^2}{2}}\right).$$

All m bits will be correct with probability greater than $\pi^m \geq \left(1 - e^{\frac{-L\epsilon'^2}{2}}\right)^m$. A probability of success greater than $\frac{1}{2}$ can be attained by taking

$$L = \frac{2}{\epsilon'^2} \ln\left(\frac{1}{1 - e^{-\frac{\ln 2}{m}}}\right) \sim \frac{512}{\delta^4(1-2u)^4}(\ln m - \ln\ln 2).$$

Hence, any efficient DET-adversary achieving a noticeable advantage against the RANDOM-HB$^{\#}$ protocol can be turned into an efficient solver of the MHB puzzle with a success probability greater than $\frac{1}{2^m} + \delta'$, where δ' is noticeable. This contradicts the assumption that LPN is hard. □

Theorem 2 (Security of RANDOM-HB$^{\#}$ in the GRS-MIM-model). *Let $\mathcal{A}$ be an adversary attacking the RANDOM-HB$^{\#}$ protocol in the GRS-MIM-model, modifying at most q executions of the protocol between an honest tag and an honest reader, running in time T, and achieving advantage greater than δ. Then, under an easily met condition on the parameter set (see the proof and Section 4.2), there is an adversary $\mathcal{A}'$ attacking the RANDOM-HB$^{\#}$ protocol in the DET-model, interacting at most q times with an honest tag, running in time $O(T)$, and impersonating a valid tag with success probability greater than $(P_{FA} + \delta)(1 - q\epsilon)$ for some negligible function ϵ. Hence, assuming the hardness of the LPN problem, the advantage of any efficient GRS-MIM-adversary against the RANDOM-HB$^{\#}$ protocol is negligible in k. As a consequence, for parameters $m = \Theta(k)$, the probability of any efficient GRS-MIM-adversary to impersonate a valid tag is negligible in k.*

Proof. As $\mathcal{A}'$ has access to an honest tag that it can query freely, there is no difficulty in simulating an honest tag to $\mathcal{A}$. The main challenge comes with the task of simulating the honest reader. Recall that in the GRS-MIM-model, the adversary is only allowed to modify the messages from the reader to the tag. $\mathcal{A}'$ launches the first phase of the adversary $\mathcal{A}$ and simulates the tag and the reader for q times as follows:

1. $\mathcal{A}'$ obtains from the real tag $\mathcal{T}_{X,Y,\eta}$ a blinding vector $\boldsymbol{b_i}$; $\mathcal{A}'$ sends $\boldsymbol{b_i}$ as the blinding vector of the simulated tag to the simulated reader.
2. $\mathcal{A}'$ sends a random vector $\boldsymbol{a_i}$ as the challenge of the simulated reader. $\mathcal{A}$ modifies it into $\boldsymbol{a'_i} = \boldsymbol{a_i} \oplus \ \ _{\boldsymbol{i}}$. $\mathcal{A}'$ forwards $\boldsymbol{a'_i}$ to the real tag.
3. The real tag returns an answer $\boldsymbol{z_i} = \boldsymbol{a'_i} \cdot X \oplus \boldsymbol{b_i} \cdot Y \oplus \ \ _{\boldsymbol{i}}$ to $\mathcal{A}'$ which uses it as the answer of the simulated tag to the simulated reader.
4. If $\ \ _{\boldsymbol{i}}$ was the all zero vector, $\mathcal{A}'$ outputs "ACCEPT" as the answer of the simulated reader, otherwise it outputs "REJECT".

After this first phase, $\mathcal{A}'$ launches the cloning phase of $\mathcal{A}$ and replicates its behaviour with the real reader. From the point of view of $\mathcal{A}$, the tag $\mathcal{T}_{X,Y,\eta}$ is perfectly simulated by $\mathcal{A}'$. Let Sim_i denote the event that the reader $\mathcal{R}_{X,Y,u}$ is correctly simulated by $\mathcal{A}$ during the i-th execution of the protocol, and Sim be the event that the reader is correctly simulated for all the q executions of the protocol, $\mathrm{Sim} = \cap_{i=1}^{q} \mathrm{Sim}_i$. Conditioning on this event Sim, the success probability of $\mathcal{A}'$ is the same as the success probability of $\mathcal{A}$, *i.e.* $P_{\mathrm{FA}} + \delta$. Hence, we have to lower bound the probability of Sim.

Consider one execution of the disturbed protocol. When $\ \ _{\boldsymbol{i}} = \boldsymbol{0}$, $\mathcal{A}'$ clearly fails at simulating the reader with a probability equal to the probability of wrongly rejecting an honest tag, *i.e.* P_{FR}. For the case $\ \ _{\boldsymbol{i}} \neq \boldsymbol{0}$ we make the following reasoning. Assume that the error vector $\ \ _{\boldsymbol{i}} \cdot X$ added by $\mathcal{A}$ has a Hamming weight d. This vector is added *before* the Bernoullian noise added by the tag, so that $\ \ _{\boldsymbol{i}}$ is independent of $\ \ _{\boldsymbol{i}} \cdot X$. Consequently, the resulting error vector $\ \ _{\boldsymbol{i}} \oplus \ \ _{\boldsymbol{i}} \cdot X$ has a Hamming weight distributed as the sum of d Bernoulli variables taking the value 1 with probability $1-\eta$ and 0 with probability η, and $m-d$ Bernoulli variables taking the value 1 with probability η and 0 with probability $1-\eta$. Hence, the mean value of the Hamming weight of the error vector is $\mu(d) = d(1-\eta) + (m-d)\eta$, and by the Chernoff bound, when $\mu(d) > t$, this weight is less than t with probability less than $e^{-\frac{(\mu-t)^2}{2\mu}}$, which remains true for any $d' \geq d$. Consequently, if the matrix X is such that for any $\ \ \neq \boldsymbol{0}$, $\mathrm{Hwt}(\ \ \cdot X)$ is high enough, outputting "REJECT" as soon as $\ \ _{\boldsymbol{i}} \neq \boldsymbol{0}$ will be a successful strategy. We formalize this as follows.

Let $d_{\min}(X) = \min_{\boldsymbol{\alpha} \neq \boldsymbol{0}} (\mathrm{Hwt}(\ \ \cdot X))$ denote the minimal distance of the matrix X. We recall the following classical result of coding theory:

Lemma 1. *Let d be an integer in $[1..\lfloor \frac{m}{2} \rfloor]$ and let H be the entropy function $H(x) = -x\log_2(x) - (1-x)\log_2(1-x)$. Then*

$$\Pr_X[d_{\min}(X) \leq d] \leq 2^{-\left(1-\frac{k_X}{m}-H(\frac{d}{m})\right)m}.$$

This is a simple consequence of the following upper bound on the number of m-bit vectors of Hamming weight less than d: $\sum_{i=0}^{d} \binom{m}{i} \leq 2^{mH(\frac{d}{m})}$. For any non-zero vector , $\cdot X$ is uniformly distributed, and hence has Hamming weight less than d with probability less than $2^{m\left(H(\frac{d}{m})-1\right)}$. The lemma follows by a union bound.

Let $\tilde{d}$ be the least integer such that $\mu > t$, *i.e.* $\tilde{d} = 1 + \left\lfloor \frac{t-\eta m}{1-2\eta} \right\rfloor$. Then for any $d \geq \tilde{d}$ when $_{\boldsymbol{i}} \neq \mathbf{0}$, one can write

$$\begin{aligned}
\Pr_{X,\boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i}] &= \Pr_{\boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i} \,|\, d_{\min}(X) > d] \cdot \Pr_X[d_{\min}(X) > d] \\
&\qquad + \Pr_{\boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i} \,|\, d_{\min}(X) \leq d] \cdot \Pr_X[\min(X) \leq d] \\
&\leq \Pr_{\boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i} \,|\, d_{\min}(X) > d] + \Pr_X[d_{\min}(X) \leq d] \\
&\leq e^{-\frac{(\mu-t)^2}{2\mu}} + 2^{-\left(1-\frac{k_X}{m}-H(\frac{d}{m})\right)m}.
\end{aligned}$$

For this upper bound to be useful, the coefficient $\left(1 - \frac{k_X}{m} - H(\frac{d}{m})\right)$ must be positive for some $d \geq \tilde{d}$, in particular for $\tilde{d}$ as it is a decreasing function of d. This is a condition which is easily met for typical values of the parameters (see Section 4.2). Note also that for the asymptotic reduction we have to define $\tilde{d}$ as the least integer such that $\mu(\tilde{d}) > (1+c)t$ for some $c > 0$ in order to ascertain that the first term in the upper bound will be negligible. This way one has, for all $d \geq \tilde{d}$, $e^{-\frac{(\mu-t)^2}{2\mu}} \leq e^{-\frac{uc^2}{2(1+c)}m}$.

Together we have $\Pr[\overline{\mathrm{Sim}_i}] \leq \epsilon$, where ϵ is a negligible function given by

$$\epsilon = \max\left\{P_{\mathrm{FR}}, \min_{d \geq \tilde{d}}\left(e^{-\frac{(\mu-t)^2}{2\mu}} + 2^{-\left(1-\frac{k_X}{m}-H(\frac{d}{m})\right)m}\right)\right\}.$$

Consequently, $\Pr[\mathrm{Sim}] \geq (1 - q\epsilon)$ and $\mathcal{A}'$ has a success probability greater than $(P_{\mathrm{FA}} + \delta)(1 - q\epsilon)$.

If δ is noticeable then $q\epsilon(P_{\mathrm{FA}} + \delta) \leq \delta/2$ for k big enough, and the success probability of $\mathcal{A}'$ is greater than $P_{\mathrm{FA}} + \frac{\delta}{2}$. This contradicts Theorem 1. □

With RANDOM-HB$^{\#}$ we have a surprisingly successful proposal. It is as computationally efficient as HB^{+} since it consists of a series of bitwise dot-product computations. At the same time it is simpler in terms of communication since there is only a single round and the total amount of data transmitted is much less than for HB^{+}. It also possesses a proof of security in the detection-based model, exactly like HB^{+}, but also against man-in-the-middle adversaries of the type used in the GRS attack. However there remains one drawback: storage. We show how to remedy this situation in the next section.

4 The Proposal HB$^{\#}$

In RANDOM-HB$^{\#}$ the tag is required to store two random $(k_X \times m)$- and $(k_Y \times m)$-binary matrices X and Y where k_X, k_Y and m are three-digit figures. The storage costs on the tag would be insurmountable. With this in mind we propose the protocol HB$^\#$. This has very modest storage requirements while preserving the computational efficiency of HB$^+$. While there are some subtle technical issues that mean we cannot transfer all the provably security results from RANDOM-HB$^\#$ to HB$^\#$ we can transfer some. These, together with a plausible conjecture, allow us to claim that HB$^\#$ is secure in the GRS-MIM-model. HB$^\#$ depends on the notion of a *Toeplitz* matrix. These were used by Krawczyk in message authentication proposals where their good distribution properties and efficient implementation were noted [14,15].

A $(k \times m)$-binary *Toeplitz* matrix M is a matrix for which the entries on every upper-left to lower-right diagonal have the same value. Since the diagonal values of a Toeplitz matrix are fixed, the entire matrix is specified by the top row and the first column. Thus a Toeplitz matrix can be stored in $k+m-1$ bits rather than the km bits required for a truly random matrix. For any $(k+m-1)$-bit vector $\boldsymbol{s}$, we denote by $T_{\boldsymbol{s}}$ the Toeplitz matrix whose top row and first column are represented by $\boldsymbol{s}$. HB$^\#$ is defined exactly as RANDOM-HB$^\#$ except that X and Y are now two random $(k_X \times m)$ and $(k_Y \times m)$-binary Toeplitz matrices.

4.1 Security Results for HB$^\#$

While there is every indication that HB$^\#$ is secure in the DET-model, this remains to be shown. A first obvious step in this direction would be to prove that the Toeplitz variant of the MHB puzzle remains hard. We state the following conjecture to stimulate further research:

Conjecture 1 (Hardness of the Toeplitz-MHB puzzle). Let k be a security parameter, $\eta \in]0, 1/2[$, and m and q be polynomials in k. Let X be a random secret $(k \times m)$-binary *Toeplitz* matrix, and $(\boldsymbol{a_1}, \ldots, \boldsymbol{a_q})$ be q random vectors of length k. Then any efficient algorithm, on input q noisy samples $(\boldsymbol{a_i}, \boldsymbol{a_i} \cdot X \oplus \ {}_{\boldsymbol{i}})$, where each bit of ${}_{\boldsymbol{i}}$ is 1 with probability η, and a random vector $\boldsymbol{a}$ of length k, outputs $\boldsymbol{z} = \boldsymbol{a} \cdot X$ with probability negligibly close to $\frac{1}{2^m}$.

Just as for RANDOM-HB$^\#$, we can relate the security of the HB$^\#$ protocol in the GRS-MIM-model to its security in the DET-model.

Theorem 3 (Security of HB$^\#$ in the GRS-MIM-model). *Let $\mathcal{A}$ be an adversary attacking the HB$^\#$ protocol in the GRS-MIM-model, modifying at most q executions of the protocol between an honest tag and an honest reader, running in time T, and achieving advantage greater than δ. Then, under an easily met condition on the parameter set (see proof of Theorem 2 and Section 4.2), there is an adversary $\mathcal{A}'$ attacking the HB$^\#$ protocol in the DET-model, interacting at most q times with an honest tag, running in time $O(T)$, and impersonating a valid tag with success probability greater than $(P_{FA}+\delta)(1-q\epsilon)$ for some negligible function ϵ.*

Proof. (Outline) The proof is analogous to that of Theorem 2 and omitted for reasons of space. It relies on the observation that Lemma 1 remains true when the probability is taken over the set of random $(k_X \times m)$-Toeplitz matrices. □

Table 2. Practical parameters for HB$^{\#}$

HB$^{\#}$ k_X	k_Y	m	η	t	False reject rate (P_{FR})	False accept rate (P_{FA})	Transmission (bits)	Storage (bits)
80	512	1164	0.25	405	2^{-45}	2^{-83}	1,756	2,918
80	512	441	0.125	113	2^{-45}	2^{-83}	1,033	1,472

Hence, the security of HB$^{\#}$ in the DET-model (which we believe to be a likely conjecture) would directly transfer to the GRS-MIM-model.

4.2 Parameter Values for HB$^{\#}$

When considering the error rates in HB$^{\#}$, we have considerable flexibility in how we set the acceptance threshold t. Recall that the false rejection rate depends on m, t, and η and the false acceptance rate depends on m and t only. The overall security of the scheme depends on k_X, k_Y and η. However, as already noted by Levieil and Fouque [16] for HB^{+}, and as is clear from the proof of Theorem 1, k_X and k_Y play two different roles: only k_Y is related to the difficulty of the LPN problem, while k_X need only be 80-bit long to achieve 80-bit security.

Some example parameters for different noise levels η are given by Levieil and Fouque [16]. These give very reasonable error rates of $P_{\mathrm{FR}} < 2^{-40}$ and $P_{\mathrm{FA}} < 2^{-80}$. When combined with the larger values of k_Y required for good security with the LPN problem, the HB$^{\#}$ protocol compares very favourably to HB^{+}. The practical characteristics are summarised in Table 2. The condition necessary for Theorems 2 and 3 to hold is verified for both sets of parameters: for the first one, $\tilde{d} = 229$ and $\left(1 - \frac{k_X}{m} - H(\frac{\tilde{d}}{m})\right) \simeq 0.216$, while for the second one $\tilde{d} = 78$ and $\left(1 - \frac{k_X}{m} - H(\frac{\tilde{d}}{m})\right) \simeq 0.145$. The storage cost of HB$^{\#}$ is $(k_X + k_Y + 2m - 2)$ bits which is larger than the $2k$ bits required for HB^{+}. However, depending on the choice of m this is not necessarily a substantial increase. The given parameter choices offer 80-bit security (using the latest results on the LPN problem), the false acceptance and rejection rates are less than 2^{-80} and 2^{-40} respectively, and the total communication requirements are around 1,500 bits. This should be compared to error rates of 2^{-1} and 2^{-20} and transmission costs of up to 80,000 bits in the case of HB^{+} (48,000 bits when $\boldsymbol{x}$ is only 80-bit long) for corresponding parameters. HB$^{\#}$ requires simple bit operations on-the-tag and thus remains computationally simple.

5 Further Work and HB$^{\#}$ Variants

General MIM adversaries. The result of Theorem 3 shows that an adversary successfully mounting an attack on HB$^{\#}$ must either (i) break HB$^{\#}$ in the DET-model (which we believe is highly improbable), or (ii) break the LPN problem, or (iii) use an undiscovered active attack involving more than manipulation of the messages from the reader. This raises the question of the security of $\mathrm{HB}^{\#}$ against general man-in-the-middle adversaries allowed to perturb any message of the protocol. Though we do not have a formal proof of such a result, we can make the following heuristic analysis. To provide an appropriate context we might recall earlier work by Krawczyk [14,15]. Let us denote by $\mathcal{H}_{\mathcal{T}}$, where $\mathcal{T}$ stands for "random Toeplitz" matrix, the (k, m)-family of k-bit to m-bit linear functions $\boldsymbol{a} \mapsto \boldsymbol{a} \cdot T_{\boldsymbol{s}}$ associated with the set of $k \times m$ binary Toeplitz matrices $T_{\boldsymbol{s}}$, each associated with a $(k+m-1)$-bit vector $\boldsymbol{s}$, and equipped with the uniform probability. The work of Krawczyk [15], which in turn references related work by Mansour *et al.* [17], in effect establishes that $\mathcal{H}_{\mathcal{T}}$ is $\frac{1}{2^m}$-balanced. In other words, for any non-zero vector $\boldsymbol{a}$, $\boldsymbol{a} \cdot T_{\boldsymbol{s}}$ is uniformly distributed over $\{0,1\}^m$. This results from the fact that if $\boldsymbol{a}$ is a non-zero vector then $\boldsymbol{a} \cdot T_{\boldsymbol{s}}$ can be re-written as the product of $\boldsymbol{s}$ with a $(k+m-1) \times m$ matrix derived from $\boldsymbol{a}$ that has rank m.

We can use this property of Toeplitz matrices to argue in favour of the resistance of $\mathrm{HB}^{\#}$ against arbitrary man-in-the-middle adversaries. Consider an attack where the adversary perturbs $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{z}$ by adding respectively three disturbance vectors , , . The modified error vector is then $'$ = ⊕ $\cdot X \oplus$ $\cdot Y \oplus$. When $\neq \mathbf{0}$ or $\neq \mathbf{0}$, then due to the $\frac{1}{2^m}$-balance of $\mathcal{H}_{\mathcal{T}}$, $'$ is uniformly distributed and the probability that modifications of the communication between tag and reader result in successful authentication is the false acceptance probability P_{FA}. The reader's decision has negligible entropy and hence yields no information on X or Y to the adversary. On the contrary, when (,) = $(\mathbf{0}, \mathbf{0})$, the answer $\boldsymbol{z}$ returned by the tag is uniformly random so that may be considered as independent of X and Y. The reader's decision depends only on ⊕ and again yields no information on X or Y to the adversary. It is helpful to note the essential difference between a man-in-the-middle attack on $\mathrm{HB}^{\#}$ and the same attack on HB^{+}. When attacking HB^{+}, *e.g.* as is done in the GRS attack, the adversary gains 1 bit of information on $\boldsymbol{x}$ at *every* tag and reader HB^{+} authentication (independently of whether it is successful or not), leading to a linear-time attack. By contrast, in the case of $\mathrm{HB}^{\#}$, whatever the strategy for choosing (, ,), the mutual information between the reader's decision and the matrices X and Y is negligible and no efficient adversary can gather noticeable information on X or Y. Though we believe that these observations can be made rigorous, it remains an open problem to extend the technique used in proof of Theorems 2 and 3 to arbitrary man-in-the-middle attacks and to find the right way of simulating the reader when the adversary can also modify $\boldsymbol{b}$ and $\boldsymbol{z}$.

Variants and optimisations. Independently of this theoretical work, there are interesting variants to $\mathrm{HB}^{\#}$ that might be of practical value. One interesting option, also mentioned in [12], is for the legitimate tag to test that the noise vector contains no more than t ones before using it. This means the probability of a false rejection would fall to zero. The main advantage of this approach would be to allow the size of m to decrease while maintaining a reasonable false acceptance rate. For instance, with $m = 256$, $\eta = 0.125$, and $t = 48$ we would ordinarily have that $P_{FA} \approx 2^{-81}$ while $P_{FR} \approx 2^{-9}$. However, this relatively high false rejection rate can be eliminated by allowing the tag to check before use.

Another possibility to decrease storage and communication costs is to reduce k_Y; for this, it might be interesting to consider the effect of using a larger noise level, *i.e.* to have $\eta > \frac{1}{4}$. In such circumstances k_Y could be reduced—while maintaining the same level of security—thereby leading to storage and communications savings. While it is not immediately clear that this would be a successful approach, when coupled with restrictions to the noise vector this may be worth exploring. Another optimisation could be to use techniques inspired by Krawczyk [14,15] to efficiently re-generate the Toeplitz matrices (*e.g.* by using a *LFSR*). We leave such proposals as topics for future research.

6 Conclusions

In this paper we have presented two new lightweight authentication protocols. While close variants of HB^{+}, these new protocols offer considerable advantages over related work in the literature. RANDOM-$HB^{\#}$ is provably secure in the detection-based model, just like HB^{+}, but it is also provably resistant to a broader class of attacks that includes [8]. The protocol $HB^{\#}$ trades some of the theoretical underpinnings to RANDOM-$HB^{\#}$ and attains a truly practical performance profile. Both RANDOM-$HB^{\#}$ and $HB^{\#}$ offer practical improvements over HB^{+}, and this remains the case even when using the problem sizes required after recent progress on solving the underlying LPN problem.

References

1. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
2. Bringer, J., Chabanne, H., Dottax, E.: HB^{++}: A Lightweight Authentication Protocol Secure Against Some Attacks. In: Proceedings of SecPerU 2006, pp. 28–33. IEEE Computer Society Press, Los Alamitos (2006)
3. Canetti, R., Halevi, S., Steiner, M.: Hardness Amplification of Weakly Verifiable Puzzles. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 17–33. Springer, Heidelberg (2005)
4. Duc, D.N., Kim, K.: Securing HB^{+} Against GRS Man-in-the-Middle Attack. In: Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security (January 23–26, 2007)
5. Feldhofer, M., Dominikus, S., Wolkerstorfer, J.: Strong Authentication for RFID Systems Using the AES Algorithm. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 357–370. Springer, Heidelberg (2004)
6. Fossorier, M.P.C., Mihaljevic, M.J., Imai, H., Cui, Y., Matsuura, K.: A Novel Algorithm for Solving the LPN Problem and its Application to Security Evaluation of the HB Protocol for RFID Authentication, `http://eprint.iacr.org/2006/197.pdf`

7. Gilbert, H., Robshaw, M.J.B., Seurin, Y.: Good Variants of HB^+ are Hard to Find. In: Proceedings of Financial Crypto 2008 (to appear)
8. Gilbert, H., Robshaw, M.J.B., Sibert, H.: An Active Attack Against HB^+: A Provably Secure Lightweight Authentication Protocol. IEE Electronics Letters 41(21), 1169–1170 (2005)
9. Girault, M., Poupard, G., Stern, J.: On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. Journal of Cryptology 19(4), 463–488 (2006)
10. Juels, A., Weis, S.A.: Authenticating Pervasive Devices With Human Protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 198–293. Springer, Heidelberg (2005)
11. Juels, A. Weis, S.A. Authenticating Pervasive Devices With Human Protocols. Version of [10] with appendices, http://saweis.net/pdfs/lpn-paper.pdf
12. Katz, J., Shin, J.: Parallel and Concurrent Security of the HB and HB^+ Protocols. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006)
13. Katz, J., Smith, A.: Analysing the HB and HB^+ Protocols in the Large Error Case, http://eprint.iacr.org/2006/326.pdf
14. Krawczyk, H.: LFSR-based Hashing and Authentication. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 129–139. Springer, Heidelberg (1994)
15. Krawczyk, H.: New Hash Functions for Message Authentication. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 301–310. Springer, Heidelberg (1995)
16. Levieil, E., Fouque, P.-A.: An Improved LPN Algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
17. Mansour, Y., Nisan, N., Tiwari, P.: The Computational Complexity of Universal Hashing. In: Proceedings of STOC 1990, pp. 235–243 (1990)
18. McLoone, M., Robshaw, M.J.B.: Public Key Cryptography and RFID. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 372–384. Springer, Heidelberg (2006)
19. Munilla, J., Peinado, A.: HB-MP: A Further Step in the HB-family of Lightweight Authentication Protocols. Computer Networks 51, 2262–2267 (2007)
20. Piramuthu, S.: HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication. CollECTeR Europe Conference (June 2006)
21. Poschmann, A., Leander, G., Schramm, K., Paar, C.: New Lightweight DES Variants Suited for RFID Applications. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
22. Shamir, A.: SQUASH - A New MAC With Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In: Proceedings of FSE 2008 (to appear)

Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle

Jens Groth[1,⋆] and Yuval Ishai[2,⋆⋆]

[1] University College London
j.groth@ucl.ac.uk
[2] Technion and University of California Los Angeles
yuvali@cs.technion.ac.il

Abstract. A shuffle of a set of ciphertexts is a new set of ciphertexts with the same plaintexts in permuted order. Shuffles of homomorphic encryptions are a key component in mix-nets, which in turn are used in protocols for anonymization and voting. Since the plaintexts are encrypted it is not directly verifiable whether a shuffle is correct, and it is often necessary to prove the correctness of a shuffle using a zero-knowledge proof or argument.

In previous zero-knowledge shuffle arguments from the literature the communication complexity grows linearly with the number of ciphertexts in the shuffle. We suggest the first practical shuffle argument with sub-linear communication complexity. Our result stems from combining previous work on shuffle arguments with ideas taken from probabilistically checkable proofs.

Keywords: Shuffle, zero-knowledge argument, sub-linear communication, homomorphic encryption, mix-net.

1 Introduction

A shuffle of ciphertexts $e_1, \ldots, e_N$ is a new set of ciphertexts $E_1, \ldots, E_N$ with the same plaintexts in permuted order. Shuffles are used in many protocols for anonymous communication and voting. It is usually important to verify the correctness of the shuffle. Take for instance a voting protocol where the ciphertexts are encrypted votes; it is important to avoid that some of the ciphertexts in the shuffle are substituted with encryptions of other votes. There has therefore been much research on designing zero-knowledge arguments[1] for the correctness of a shuffle [37,1,2,17,30,31,21,16,33,34,32,15,24,38].

When designing shuffle arguments, efficiency is a major concern. It is realistic to have elections with millions of encrypted votes, in which case the statement to be proven is very large. In this paper, our main goal is to get a *practical* shuffle argument with low

⋆ Research done in part while visiting IPAM. Part of work done while at UCLA supported by NSF ITR/Cybertrust grant 0456717.

⋆⋆ Research done in part while visiting IPAM. Supported by BSF grant 2004361, ISF grant 1310/06, and NSF grants 0205594, 0430254, 0456717, 0627781, 0716835, 0716389.

[1] By zero-knowledge *arguments* [8] we refer to computationally-sound zero-knowledge proofs [20].

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 379–396, 2008.

communication complexity. A theoretical solution to this problem would be to use Kilian's communication-efficient zero-knowledge argument [26] (see also Micali [29]). This method, however, requires a reduction to Circuit Satisfiability, a subsequent application of the PCP-theorem [4,3,12], and using a collision-free hash-function to build a hash-tree that includes the entire PCP. Even with the best PCP constructions known to date (cf. [7]), such an approach would be inefficient in practice.

OUR CONTRIBUTION. We present a sublinear-communication 7-move public coin perfect zero-knowledge argument of knowledge for the correctness of a shuffle of ElGamal ciphertexts [13]. (The protocol is presented in the common random string model, but can also be implemented in the plain model at the cost of a slightly higher constant number of rounds.) All shuffle arguments previously suggested in the literature have communication complexity $\Omega(N)\kappa$, where N is the number of ciphertexts in the shuffle and κ is a security parameter specifying the finite group over which the scheme works. Our shuffle argument has communication complexity $O(m^2 + n)\kappa$ for m and n such that $N = mn$. (The constant in the expression is low as well, see Section 8 for a more precise efficiency analysis.) With $m = N^{1/3}$ this would give a size of $O(N^{2/3})\kappa$ bits, but in practice a smaller choice of m will usually be better for computational reasons. Our shuffle argument moderately increases the prover's computational burden and reduces the amount of communication and the verifier's computational burden in comparison with previous work.

For practical purposes it will be natural to use the Fiat-Shamir heuristic [14] (i.e. compute the verifier's public-coin challenges using a cryptographic hash-function) to make our shuffle argument non-interactive. The Fiat-Shamir heuristic justifies reducing the communication and verifier computation at the cost of increased prover computation, since the non-interactive shuffle argument needs to be computed only once by the prover but may be distributed to and checked by many verifiers. Letting the prover do some extra work in order to reduce the communication and the computational burden of each verifier is therefore a good trade-off in practice. To the best of our knowledge, our protocol is the first practical instance of a sublinear-communication argument for any interesting nontrivial statement.

We have some further remarks on our result. Our technique also applies to other homomorphic cryptosystems, for instance Paillier encryption [35]; a more general treatment of a wider class of homomorphic encryptions can be obtained along the lines of [21]. For simplicity we focus just on ElGamal encryption in this paper. Similarly to previous shuffle arguments from the literature, we will present our protocol as an *honest verifier* zero-knowledge argument. There are very efficient standard techniques for converting honest verifier zero-knowledge arguments into fully zero-knowledge arguments [10,18,22].

TECHNIQUES. Our starting point is the honest verifier zero-knowledge shuffle argument by Groth [21], which builds on ideas by Neff [30]. Borrowing some of the ideas underlying the PCP theorem, namely the use of Hadamard codes and batch-verification techniques, we reduce the size of the shuffle argument. We note that unlike Kilian [26] we do not reduce the shuffle statement to an NP-complete language such as SAT; instead we work directly with the ciphertexts in the shuffle statement. Moreover, while we use ideas behind the PCP theorem we do not make use of a full-blown PCP.

In particular, our argument avoids any use of linearity testing, low-degree testing, or other forms of code proximity testing that appear in all known PCPs.

RELATED WORK. Our work was inspired by the recent work of Ishai, Kushilevitz, and Ostrovsky [25], which introduced an approach for constructing sublinear-communication arguments using exponentially long but succinctly described PCPs. Similarly to [25] we use short *homomorphic* commitments as the main cryptographic building block. There are, however, several important differences between our techniques and those from [25]. In particular, the arguments obtained in [25] do not address our *zero-knowledge* requirement (and are only concerned with soundness), they inherently require the verifier to use *private coins* (which are undesirable in the context of our application), and they employ *linearity testing* that subsequently requires soundness amplification. Finally, the approach of [25] is generic and does not account for the special structure of the shuffle problem; this structure is crucial for avoiding an expensive reduction to SAT.

2 Preliminaries

2.1 Notation

We let Σ_N denote the symmetric group on $\{1, 2, \ldots, N\}$. Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(\kappa) \approx g(\kappa)$ when $|f(\kappa) - g(\kappa)| = O(\kappa^{-c})$ for every constant c. We say that the function f is *negligible* when $f(\kappa) \approx 0$ and that it is *overwhelming* when $f(\kappa) \approx 1$.

Algorithms in our shuffle argument will get a security parameter κ as input, which specifies the size of the group we are working over. Sometimes we for notational simplicity avoid writing this explicitly, assuming κ can be deduced indirectly from other inputs given to the algorithms.

All our algorithms will be probabilistic polynomial time algorithms. We will assume that they can sample randomness from sets of the type $\mathbb{Z}_q$. We note that such randomness can be sampled from a source of uniform random bits in expected polynomial time (in $\log q$).

We write $A(x; r) = y$ when A, on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y := A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S.

When defining security, we assume that there is an adversary attacking our scheme. This adversary is modeled as a non-uniform polynomial time stateful algorithm. By stateful, we mean that we do not need to give it the same input twice, it remembers from the last invocation what its state was. This makes the notation a little simpler, since we do not need to explicitly write out the transfer of state from one invocation to the next.

2.2 Group Generation

We will work over a group G_q of a prime order q. This could for instance be a subgroup of $\mathbb{Z}_p^*$, where p is a prime and $\gcd(q^2, p-1) = q$; or it could be an elliptic curve group or subgroup. We will assume the discrete logarithm problem is hard in G_q. More precisely, let $\mathcal{G}$ be a generating algorithm that takes a security parameter κ as input and outputs $gk := (q, G_q, g)$, where by G_q we denote a computationally efficient representation of the group and g is a random generator for G_q. The discrete logarithm assumption says that for any non-uniform polynomial time adversary $\mathcal{A}$:

$$\Pr\left[(q, G_q, g) \leftarrow \mathcal{G}(1^\kappa); x \leftarrow \mathbb{Z}_q; h := g^x : \mathcal{A}(q, G_q, g, h) = x\right] \approx 0.$$

(When the randomness of $\mathcal{G}$ is taken from a common random string, the above definition needs to be strengthened so that $\mathcal{A}$ is given the randomness used by $\mathcal{G}$.)

2.3 Generalized Pedersen Commitment

We will use a variant of the Pedersen commitment scheme [36] that permits making a commitment to a length-n vector in $\mathbb{Z}_q^n$ rather than a single element of $\mathbb{Z}_q$ as in Pedersen's original commitment. A crucial feature of this generalization is that the amount of communication it involves does not grow with n. The generalized scheme proceeds as follows. The key generation algorithm K_{com} takes (q, G_q, g) as input and outputs a commitment key $ck := (g_1, \ldots, g_n, h)$, where $g_1, \ldots, g_n, h$ are randomly chosen generators of G_q. The message space is $\mathcal{M}_{ck} := \mathbb{Z}_q^n$, the randomizer space is $\mathcal{R}_{ck} := \mathbb{Z}_q$ and the commitment space is $\mathcal{C}_{ck} := G_q$. (The parameter n will be given as an additional input to all algorithms; however, we prefer to keep it implicit in the notation.)

To commit to an n-tuple $(m_1, \ldots, m_n) \in \mathbb{Z}_q^n$ we pick randomness $r \leftarrow \mathbb{Z}_q$ and compute the commitment $C := h^r \prod_{i=1}^n g_i^{m_i}$. The commitment is perfectly hiding since no matter what the messages are, the commitment is uniformly distributed in G_q. The commitment is computationally binding under the discrete logarithm assumption; we will skip the simple proof.

The commitment key ck will be part of the common random string in our shuffle argument. We remark that it can be sampled from a random string. We write $C := \text{com}_{ck}(m_1, \ldots, m_n; r)$ for making a commitment to $m_1, \ldots, m_n$ using randomness r. The commitment scheme is homomorphic, i.e., for all $m_1, m_1', \ldots, m_n, m_n', r, r' \in Z_q$ we have

$$\text{com}_{ck}(m_1, \ldots, m_n; r) \cdot \text{com}_{ck}(m_1', \ldots, m_n'; r') = \text{com}_{ck}(m_1 + m_1', \ldots, m_m + m_n'; r + r').$$

In some cases we will commit to less than n elements; this can be accomplished quite easily by setting the remaining messages to 0.

We will always assume that parties check that commitments are valid, meaning they check that $C \in G_q$. If G_q is a subgroup of $\mathbb{Z}_p^*$ this can be done by checking that $C^q = 1$, however, batch verification techniques can be used to lower this cost when we have multiple commitments to check.[2] If G_q is an elliptic curve of order q, then the validity check just consists of checking that C is a point on the curve, which is very inexpensive.

[2] See also [21] for a variant of the Pedersen commitment scheme over $\mathbb{Z}_p^*$ that makes it possible to completely eliminate the cost of verifying validity.

2.4 ElGamal Encryption

ElGamal encryption [13] in the group G_q works as follows. The public key is $pk := y = g^x$ with a random secret key $sk := x \leftarrow \mathbb{Z}_q^*$. The message space is $\mathcal{M}_{pk} := G_q$, the randomizer space is $\mathcal{R}_{pk} := \mathbb{Z}_q$ and the ciphertext space is $\mathcal{C}_{pk} := G_q \times G_q$. To encrypt a message $m \in G_q$ using randomness $R \in \mathbb{Z}_q$ we compute the ciphertext $E_{pk}(m; R) := (g^R, y^R m)$. To decrypt a ciphertext (u, v) we compute $m = vu^{-x}$.

The semantic security of ElGamal encryption is equivalent to the DDH assumption. Semantic security may be needed for the shuffle itself to be secure; however, the security of our shuffle argument will rely on the discrete logarithm assumption only. In particular, our shuffle argument is still sound and zero-knowledge even if the cryptosystem is insecure or the decryption key has been exposed.

ElGamal encryption is homomorphic with entry-wise multiplication in the ciphertext space. For all $(m, R), (m', R') \in \mathcal{M}_{pk} \times \mathcal{R}_{pk}$ we have

$$E_{pk}(mm'; R + R') = (g^{R+R'}, y^{R+R'} mm')$$
$$= (g^R, y^R m) \cdot (g^{R'}, y^{R'} m') = E_{pk}(m; R) \cdot E_{pk}(m'; R').$$

We will always assume that the ciphertexts in the shuffle are valid, i.e., $(u, v) \in G_q \times G_q$. Batch verification techniques can reduce the cost of verifying validity when we have multiple ciphertexts. To further reduce the cost of ciphertext verification, Groth [21] suggests a variant of ElGamal encryption that makes batch-checking ciphertext validity faster. Our shuffle argument works also for this variant of ElGamal encryption.

Our shuffle argument works with many types of cryptosystems; the choice of ElGamal encryption is made mostly for notational convenience. Our technique can be directly applied with any homomorphic cryptosystem that has a message space of order q. We are neither restricted to using the same underlying group (q, G_q, g) as the commitment scheme nor restricted to using ElGamal encryption or variants thereof. Using techniques from [21] it is also possible to generalize the shuffle argument to work for cryptosystems that do not have message spaces of order q. This latter application does require a few changes to the shuffle argument though and does increase the complexity of the shuffle argument, but the resulting protocol still has the same sub-linear asymptotic complexity.

2.5 Special Honest Verifier Zero-Knowledge Arguments of Knowledge

We will assume there is a setup algorithm $\mathcal{G}$ that generates some setup information gk. This setup information could for instance be a description of a group that we will be working in. Consider a pair of probabilistic polynomial time interactive algorithms (P, V) called the prover and the verifier. They may have access to a common random string σ generated by a probabilistic polynomial time key generation algorithm K. We consider a polynomial time decidable ternary relation R. For an element x we call w a witness if $(gk, x, w) \in R$. We define a corresponding group-dependent language L_{gk} consisting of elements x that have a witness w such that $(gk, x, w) \in R$. We write $\mathrm{tr} \leftarrow \langle P(x), V(y) \rangle$ for the public transcript produced by P and V when interacting on inputs x and y together with the randomness used by V. This transcript ends with V either accepting or rejecting. We sometimes shorten the notation by saying $\langle P(x), V(y)\rangle = b$ if V ends by accepting, $b = 1$, or rejecting, $b = 0$.

Definition 1 (Argument). *The triple (K, P, V) is called an* argument *for relation R with setup $\mathcal{G}$ if for all non-uniform polynomial time interactive adversaries $\mathcal{A}$ we have*

Completeness

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^\kappa); \sigma \leftarrow K(gk); (x,w) \leftarrow \mathcal{A}(gk,\sigma) : \\ (gk,x,w) \notin R \text{ or } \langle P(gk,\sigma,x,w), V(gk,\sigma,x)\rangle = 1\Big] \approx 1.$$

Computational soundness

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^\kappa); \sigma \leftarrow K(gk); x \leftarrow \mathcal{A}(gk,\sigma) : \\ x \notin L_{gk} \text{ and } \langle \mathcal{A}, V(gk,\sigma,x)\rangle = 1\Big] \approx 0.$$

Definition 2 (Public coin argument). *An argument (K, P, V) is* public coin *if the verifier's messages are chosen uniformly at random independently of the messages sent by the prover and the setup parameters gk, σ.*

We define special honest verifier zero-knowledge (SHVZK) [9] for a public coin argument as the ability to simulate the transcript for any set of challenges without access to the witness.

Definition 3 (Perfect special honest verifier zero-knowledge). *The public coin argument (K, P, V) is called a special honest verifier zero-knowledge argument for R with setup $\mathcal{G}$ if there exists a probabilistic polynomial time simulator S such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^\kappa); \sigma \leftarrow K(gk); (x,w,\rho) \leftarrow \mathcal{A}(gk,\sigma); \\ \mathrm{tr} \leftarrow \langle P(gk,\sigma,x,w), V(gk,\sigma,x;\rho)\rangle : (gk,x,w) \in R \text{ and } \mathcal{A}(\mathrm{tr}) = 1\Big] \\ = \Pr\Big[gk \leftarrow \mathcal{G}(1^\kappa); \sigma \leftarrow K(gk); (x,w,\rho) \leftarrow \mathcal{A}(gk,\sigma); \\ \mathrm{tr} \leftarrow S(gk,\sigma,x,\rho) : (gk,x,w) \in R \text{ and } \mathcal{A}(\mathrm{tr}) = 1\Big].$$

We remark that there are efficient techniques to convert SHVZK arguments into zero-knowledge arguments for arbitrary verifiers in the common random string model [10,18,22]. In this paper, we will therefore for simplicity focus just on the special honest verifier zero-knowledge case.

WITNESS-EXTENDED EMULATION. We shall define an argument of knowledge[3] through witness-extended emulation, the name taken from Lindell [28]. Whereas Lindell's definition pertains to proofs of knowledge in the plain model, we will adapt his

[3] The standard definition of *proofs* of knowledge by Bellare and Goldreich [5] does not apply in our setting, since we work in the common random string model and are interested in *arguments* of knowledge. See Damgård and Fujisaki [11] for a discussion of this issue.

definition to the setting of public coin arguments in the common random string model. Informally, our definition says: given an adversary that produces an acceptable argument with probability ϵ, there exists an emulator that produces a similar argument with probability ϵ, but at the same time provides a witness.

Definition 4 (Witness-extended emulation). *We say the public coin argument (K, P, V) has witness-extended emulation if for all deterministic polynomial time P^* there exists an expected polynomial time emulator E such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\begin{aligned}
&\Pr\Big[gk \leftarrow \mathcal{G}(1^\kappa); \sigma \leftarrow K(gk); (x, s) \leftarrow \mathcal{A}(gk, \sigma);\\
&\qquad \mathrm{tr} \leftarrow \langle P^*(gk, \sigma, x, s), V(gk, \sigma, x)\rangle : \mathcal{A}(\mathrm{tr}) = 1\Big]\\
\approx &\Pr\Big[gk \leftarrow \mathcal{G}(1^\kappa); \sigma \leftarrow K(gk); (x, s) \leftarrow \mathcal{A}(gk, \sigma);\\
&\qquad (\mathrm{tr}, w) \leftarrow E^{\langle P\ (gk,\sigma,x,s),V(gk,\sigma,x)\rangle}(gk, \sigma, x) :\\
&\qquad \mathcal{A}(\mathrm{tr}) = 1 \text{ and if } \mathrm{tr} \text{ is accepting then } (gk, x, w) \in R\Big],
\end{aligned}$$

where E has access to a transcript oracle $\langle P^*(gk, \sigma, x, s), V(gk, \sigma, x)\rangle$ that can be rewound to a particular round and run again with the verifier using fresh randomness.

We think of s as being the state of P^*, including the randomness. Then we have an argument of knowledge in the sense that the emulator can extract a witness whenever P^* is able to make a convincing argument. This shows that the definition implies soundness. We remark that the verifier's randomness is part of the transcript and the prover is deterministic. So combining the emulated transcript with gk, σ, x, s gives us the view of both the prover and the verifier and at the same time gives us the witness.

Damgård and Fujisaki [11] have suggested an alternative definition of an argument of knowledge in the presence of a common random string. Witness-extended emulation as defined above implies knowledge soundness as defined by them [22].

THE FIAT-SHAMIR HEURISTIC. The Fiat-Shamir heuristic [14] can be used to make public coin SHVZK arguments non-interactive. In the Fiat-Shamir heuristic the verifier's challenges are computed by applying a cryptographic hash-function to the transcript of the protocol. Security can be formally argued in the random oracle model [6], in which the hash-function is modeled as a completely random function that returns a random string on each input it has not been queried before. While the Fiat-Shamir heuristic is not sound in general [19], it is still commonly believed to be a safe practice when applied to "natural" protocols.

2.6 Problem Specification and Setup

We will construct a 7-move public coin perfect SHVZK argument for the relation

$$\begin{aligned}
R = \Big\{&(gk = (q, G_q, g), (pk = y, e_1, \ldots, e_N, E_1, \ldots, E_N), (\pi, R_1, \ldots, R_N)) \;\Big|\\
&y \in G_q \wedge \pi \in \Sigma_N \wedge R_1, \ldots, R_N \in \mathcal{R}_{pk} \wedge \forall i : E_i = e_{\pi^{-1}(i)} E_{pk}(1; R_i)\Big\}.
\end{aligned}$$

In our SHVZK argument, the common random string σ will be generated as a public key $(g_1, \ldots, g_n, h)$ for the n-element Pedersen commitment scheme described in Section 2.3. Depending on the applications, there are many possible choices for who generates the commitment key and how this generation is done. For use in a mix-net, we could for instance imagine that there is a setup phase, where the mix-servers run a multi-party computation protocol to generate the setup and the commitment key. Another option is to let the verifier generate the common random string, since it is easy to verify whether a commitment key is valid or not. This option yields an 8-move (honest-verifier zero-knowledge) argument in the plain model.[4]

2.7 Polynomial Identity Testing

For completeness we state a variation of the well-known Schwartz-Zippel lemma that we use several times in the paper.

Lemma 1 (Schwartz-Zippel). *Let p be a non-zero multivariate polynomial of degree d over $\mathbb{Z}_q$, then the probability of $p(x_1, \ldots, x_\nu) = 0$ for randomly chosen $x_1, \ldots, x_\nu \leftarrow \mathbb{Z}_q$ is at most d/q.*

The Schwartz-Zippel lemma is frequently used in polynomial identity testing. Given two multi-variate polynomials p_1 and p_2 we can test whether $p_1(x_1, \ldots, x_\nu) - p_2(x_1, \ldots, x_\nu) = 0$ for random $x_1, \ldots, x_\nu \leftarrow \mathbb{Z}_q$. If the two polynomials are identical this will always be true, whereas if the two polynomials are different then there is only probability $\max(d_1, d_2)/q$ for the equality to hold.

3 Product of Committed Elements

Consider a sequence of commitments $A_1, \ldots, A_m$ and a value $a \in \mathbb{Z}_q$. We will give an SHVZK argument of knowledge of $\{a_{ij}\}_{i=1,j=1}^{m,n}$ and $\{r_i\}_{i=1}^{m}$ such that

$$\begin{aligned} A_1 &= \mathrm{com}_{ck}(a_{11}, a_{12}, \ldots, a_{1n}; r_1) \\ &\vdots \\ A_m &= \mathrm{com}_{ck}(a_{m1}, a_{m2}, \ldots, a_{mn}; r_m) \end{aligned} \qquad \text{and} \qquad a = \prod_{i=1}^{m}\prod_{j=1}^{n} a_{ij} \mod q.$$

The argument is of sub-linear size; the prover will send m^2 commitments and $2n$ elements from $\mathbb{Z}_q$, where $N = mn$ is the total number of committed elements a_{ij}. For $m = N^{1/3}$ this gives a size of $O(N^{2/3})\kappa$ bits.

The argument is quite complex so let us first describe some of the ideas that go into it. In our argument, the prover will prove knowledge of the contents of the commitments.

[4] We can also get full zero-knowledge in the plain model. The verifier picks the common random string as above and also picks an additional key for a trapdoor commitment scheme. The verifier then makes engages in a zero-knowledge proof of knowledge of the trapdoor. We can now use the standard techniques for converting honest verifier zero-knowledge arguments to full zero-knowledge arguments [10,18,22]. By running the two proofs in parallel, the round complexity is only 8. Note, however, that since the verifier must know the secret trapdoor of the additional commitment scheme, the protocol is no longer public coin.

For the sake of simplicity we will first describe the argument assuming the prover knows the contents of the commitments and by the computational binding property of the commitment scheme is bound to these values. We will also for the sake of simplicity just focus on soundness and later when giving the full protocol add extra parts that will give us honest verifier zero-knowledge and witness-extended emulation. (Note that even completeness and soundness alone are nontrivial to achieve when considering *sublinear communication* arguments.)

Consider first commitments $A_1, \ldots, A_m$ as described above. The verifier will pick a random challenge $s_1, \ldots, s_m$. By the homomorphic property

$$\prod_{i=1}^{m} A_i^{s_i} = \mathrm{com}_{ck}(\sum_{i=1}^{m} s_i a_{i1}, \ldots, \sum_{i=1}^{m} s_i a_{in}; \sum_{i=1}^{m} s_i r_i).$$

In our argument the prover will open this commitment multi-exponentiation as $f_1 := \sum_{i=1}^{m} s_i a_{i1}, \ldots, f_n := \sum_{i=1}^{m} s_i a_{in}, z := \sum_{i=1}^{m} s_i r_i$.

Consider now the case where we have three sets of commitments $\{A_i\}_{i=1}^m, \{B_\ell\}_{\ell=1}^m, \{C_{i\ell}\}_{i=1,\ell=1}^{m,m}$ containing respectively $m \times n$ matrices A, B and $m^2 \times n$ matrix C. The verifier will choose random challenges $s_1, \ldots, s_m, t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$. The prover can open the commitment products $\prod_{i=1}^m A_i^{s_i}$, $\prod_{\ell=1}^m B_\ell^{t_\ell}$, $\prod_{i=1}^m \prod_{\ell=1}^m C_{i\ell}^{s_i t_\ell}$ as described above. This gives us for each of the n columns

$$f_j := \sum_{i=1}^{m} s_i a_{ij} \quad , \quad F_j := \sum_{\ell=1}^{m} t_\ell b_{\ell j} \quad , \quad \phi_j := \sum_{i=1}^{m} \sum_{\ell=1}^{m} s_i t_\ell c_{i\ell j}.$$

In our proofs the verifier will check for each column that $\phi_j = f_j F_j$. These checks can be seen as quadratic equations in variables $s_1, \ldots, s_m, t_1, \ldots, t_m$ of the form

$$(\sum_{i=1}^{m} s_i a_{ij})(\sum_{\ell=1}^{m} t_\ell b_{\ell j}) = \sum_{i=1}^{m} \sum_{\ell=1}^{m} s_i t_\ell c_{i\ell j}.$$

If $c_{i\ell j} = a_{ij} b_{\ell j}$ for all i, ℓ, j the check will always pass, whereas if this is not the case, then by the Schwartz-Zippel lemma there is overwhelming probability over the choice of $s_1, \ldots, s_m, t_1, \ldots, t_m$ that the check will fail. (This type of checking is also used in the Hadamard-based PCP of Arora et al. [3].) We therefore have an argument for C_{ii} being a commitment to $\{a_{ij} b_{ij}\}_{j=1}^n$. The commitments $C_{i\ell}$ for $i \neq \ell$ are just fillers that make the argument work, we will not need them for anything else. In the argument we only reveal $O(n)$ elements in $\mathbb{Z}_q$ to simultaneously prove $N = mn$ equalities $c_{iij} = a_{ij} b_{ij}$; this is what will give us sub-linear communication complexity.

Let us now explain how we choose the matrix B. For $1 \leq I \leq m, 1 \leq J \leq n$ we set $b_{IJ} := \prod_{i=1}^{I-1} \prod_{j=1}^{n} a_{ij} \cdot \prod_{j=1}^{J} a_{Ij}$. This means that B is a matrix chosen such that b_{ij} is the previous element in the matrix B multiplied with a_{ij}. In particular, we have $b_{mn} = \prod_{i=1}^{m} \prod_{j=1}^{n} a_{ij} = a$. In addition, we will have an extra column with $b_{10} := 1$ and for $1 < i \leq m : b_{i0} := b_{i-1,n}$. In other words, the 0th column vector is the nth column vector of B shifted one step down. The prover will make a separate set of m commitments $B'_1, \ldots, B'_m$ to this column. Choosing $B'_1 := \mathrm{com}_{ck}(1; 0)$ it is straightforward to verify that $b_{10} = 1$. To show that the rest of the 0th column is correctly constructed the prover will open $\prod_{\ell=2}^{m-1}(B'_\ell)^{t_{\ell-1}}$ to the message $F_n - t_m a$. The linear equations give us $\sum_{\ell=2}^{m-1} t_{\ell-1}b_{\ell 0} + t_m a = \sum_{\ell=1}^{m} t_\ell b_{\ell n}$, which by the Schwartz-Zippel lemma has negligible probability of being true unless $b_{mn} = a$ and $b_{\ell+1,0} = b_{\ell n}$ for $1 \leq \ell < m$.

We have now described B extended with a 0th column vector. Write $\widetilde{B}$ for the matrix with the 0th column and the first $n-1$ columns of B. We will apply the A, B, C matrix argument we described before to the matrices $A, \widetilde{B}, C$, where we use commitments $C_{ii} := B_i$. This argument demonstrates for each $1 \leq j \leq n$ that $b_{ij} = a_{ij}b_{i,j-1}$. Putting everything together we now have: $b_{10} = 1, b_{ij} = a_{ij}b_{i,j-1}, b_{i0} = b_{i-1,n}$ and $b_{mn} = a$, which is sufficient to conclude that $a = \prod_{i=1}^{m}\prod_{j=1}^{n} a_{ij}$.

We will now describe the full protocol. The most significant change from the description given above is that we now add also elements a_{0j}, b_{0j} that are chosen at random to the matrices. The role of these elements is to give honest verifier zero-knowledge. The prover reveals elements of the form $f_j := a_{0j} + \sum_{i=1}^{m} s_i a_{ij}$ and $F_j := b_{0j} + \sum_{\ell=1}^{m} t_\ell b_{\ell j}$, which reveal nothing about $\sum_{i=1}^{m} s_i a_{ij}$ and $\sum_{\ell=1}^{m} t_\ell b_{\ell j}$ when a_{0j} and b_{0j} are random.

Initial message

$a_{01}, \ldots, a_{0n} \leftarrow \mathbb{Z}_q$; $r_0 \leftarrow \mathcal{R}_{ck}$; $A_0 := \mathrm{com}_{ck}(a_{01}, a_{02}, \ldots, a_{0n}; r_0)$

For $1 \leq I \leq m, 1 \leq J \leq n$: $b_{IJ} := \prod_{i=1}^{I-1}\prod_{j=1}^{n} a_{ij} \cdot \prod_{j=1}^{J} a_{Ij}$

$b_{01}, \ldots, b_{0n} \leftarrow \mathbb{Z}_q$; $r_{b0}, r_{b1} \ldots, r_{bm} \leftarrow \mathcal{R}_{ck}$

$$\begin{aligned} B_0 &:= \mathrm{com}_{ck}(b_{01}, b_{02}, \ldots, b_{0n}; r_{b0}) \\ B_1 &:= \mathrm{com}_{ck}(b_{11}, b_{12}, \ldots, b_{1n}; r_{b1}) \\ &\vdots \\ B_m &:= \mathrm{com}_{ck}(b_{m1}, b_{m2}, \ldots, b_{mn}; r_{bm}) \end{aligned}$$

Define $b_{10} := 1, b_{20} := b_{1n}, \ldots, b_{m0} := b_{m-1,n}$

$r'_2, \ldots, r'_m \leftarrow \mathcal{R}_{ck}$; $B'_2 := \mathrm{com}_{ck}(b_{20}; r'_2), \ldots, B'_m := \mathrm{com}_{ck}(b_{m0}; r'_m)$

$b_{00} \leftarrow \mathbb{Z}_q$; $r'_0 \leftarrow \mathcal{R}_{ck}$; $B'_0 := \mathrm{com}_{ck}(b_{00}; r'_0)$

$\hat{r} \leftarrow \mathcal{R}_{ck}$; $\widehat{B} := \mathrm{com}_{ck}(b_{0n}; \hat{r})$

For $0 \leq i, \ell \leq m$: $r_{i\ell} \leftarrow \mathcal{R}_{ck}$ and for $1 \leq i \leq m$: $r_{ii} := r_{bi}$.

For $0 \leq i, \ell \leq m$:

$$C_{i\ell} := \mathrm{com}_{ck}(a_{i1}b_{\ell 0}, \ldots, a_{in}b_{\ell,n-1}; r_{i\ell})$$

Since $b_{ij} = a_{ij}b_{i,j-1}$ and $r_{ii} = r_{bi}$ we have for $1 \leq i \leq m$ that $C_{ii} = B_i$.

Send $(A_0, B_0, B'_0, B'_2, \ldots, B'_m, \widehat{B}, C_{00}, \ldots, C_{mm})$ to the verifier

Challenge: $s_1, \ldots, s_m, t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$

Answer

For $1 \leq j \leq n$: $f_j := a_{0j} + \sum_{i=1}^{m} s_i a_{ij}$; $F_j := b_{0j} + \sum_{\ell=1}^{m} t_\ell b_{\ell j}$; $F_0 := b_{00} + \sum_{\ell=1}^{m} t_\ell b_{\ell 0}$

$z := r_0 + \sum_{i=1}^{m} s_i r_i$; $z_b := r_{b0} + \sum_{\ell=1}^{m} t_\ell r_{b\ell}$; $z' := r'_0 + \sum_{\ell=2}^{m} t_\ell r'_\ell$; $\hat{z} := \hat{r} + \sum_{\ell=2}^{m} t_{\ell-1} r'_\ell$

$z_{ab} := r_{00} + \sum_{i=1}^{m} s_i r_{i0} + \sum_{\ell=1}^{m} t_\ell r_{0\ell} + \sum_{i=1}^{m}\sum_{\ell=1}^{m} s_i t_\ell r_{i\ell}$

Send $(f_1, \ldots, f_n, F_0, \ldots, F_n, z, z_b, z', \hat{z}, z_{ab})$ to the verifier

Verification

Check $A_0 \prod_{i=1}^{m} A_i^{s_i} = \mathrm{com}_{ck}(f_1, \ldots, f_n; z)$
For $1 \leq \ell \leq m$ set $B_\ell := c_{\ell\ell}$. Check $B_0 \prod_{\ell=1}^{m} B_\ell^{t_\ell} = \mathrm{com}_{ck}(F_1, \ldots, F_n; z_b)$
Set $B'_1 := \mathrm{com}_{ck}(1; 0)$. Check $B'_0 \prod_{\ell=1}^{m} (B'_\ell)^{t_\ell} = \mathrm{com}_{ck}(F_0; z')$.
Check $\hat{B} \prod_{\ell=2}^{m} (B'_\ell)^{t_{\ell-1}} = \mathrm{com}_{ck}(F_n - t_m a; \hat{z})$
Check

$$C_{00} \cdot \prod_{i=1}^{m} C_{i0}^{s_i} \cdot \prod_{\ell=1}^{m} C_{0\ell}^{t_\ell} \cdot \prod_{i=1}^{m} \prod_{\ell=1}^{m} C_{i\ell}^{s_i t_\ell} = \mathrm{com}_{ck}(f_1 F_0, \ldots, f_n F_{n-1}; z_{ab})$$

Theorem 1. *The protocol described above is a 3-move public-coin perfect SHVZK argument of knowledge of a_{ij} and r_i such that $a = \prod_{i=1}^{m} \prod_{j=1}^{n} a_{ij}$ and for all i we have $A_i = \mathrm{com}_{ck}(a_{i1}, \ldots, a_{in}; r_i)$.*

The proof can be found in the full paper [23].

4 Committed Permutation of Known Elements

Consider a vector of commitments $B_1, \ldots, B_m$ and a set of values $\{a_{ij}\}_{i=1,j=1}^{m,n}$. In this section we will give an argument of knowledge of $\pi \in \Sigma_N$ and $\{r_i\}_{i=1}^{m}$ such that:

$$\begin{aligned} B_1 &= \mathrm{com}_{ck}(a_{\pi^{-1}(11)}, a_{\pi^{-1}(12)}, \ldots, a_{\pi^{-1}(1n)}; r_1) \\ &\vdots \\ B_m &= \mathrm{com}_{ck}(a_{\pi^{-1}(m1)}, a_{\pi^{-1}(m2)}, \ldots, a_{\pi^{-1}(mn)}; r_m) \end{aligned}$$

(Here we identify $[N]$ with $[m] \times [n]$.)

Our argument uses Neff's idea [30], which is to let the verifier pick a value x at random and let the prover argue that the committed values b_{ij} satisfy $\prod_{i=1}^{m} \prod_{j=1}^{n} (x - b_{ij}) = \prod_{i=1}^{m} \prod_{j=1}^{n} (x - a_{ij})$. If the committed b_{ij} are a permutation of a_{ij} this equation holds, since polynomials are invariant under permutation of their roots. On the other hand, if b_{ij} are not a permutation of a_{ij}, then by the Schwartz-Zippel lemma there is negligible chance over the choice of x for the equality to hold.

Initial challenge: $x \leftarrow \mathbb{Z}_q$
Answer: Define $B'_1 := \mathrm{com}_{ck}(x, \ldots, x; 0) B_1^{-1}, \ldots, B'_m := \mathrm{com}_{ck}(x, \ldots, x; 0) B_m^{-1}$ and $a := \prod_{i=1}^{m} \prod_{j=1}^{n} (x - a_{ij})$.
Make a 3-move argument of knowledge of openings of $B'_1, \ldots, B'_m$ such that the product of all the entries is a.

Theorem 2. *The protocol is a 4-move public coin perfect SHVZK argument of knowledge of a_{ij}, r_i, π such that $B_i := \mathrm{com}_{ck}(a_{\pi^{-1}(i1)}, \ldots, a_{\pi^{-1}(in)}; r_i)$.*

We refer to the full paper [23] for a proof.

5 Multi-exponentiation to Committed Exponents

Consider a set of commitments $A_1, \ldots, A_m$, a matrix of ciphertexts $E_{11}, \ldots, E_{mn}$ and a ciphertext E. In this section we will give an argument of knowledge of $\{a_{ij}\}_{i=1,j=1}^{m,n}$, $\{r_i\}_{i=1}^m$ and R such that:

$$\begin{aligned} A_1 &= \mathrm{com}_{ck}(a_{11}, a_{12}, \ldots, a_{1n}; r_1) \\ &\vdots \\ A_m &= \mathrm{com}_{ck}(a_{m1}, a_{m2}, \ldots, a_{mn}; r_m) \end{aligned} \quad \text{and} \quad E = E_{pk}(1;R) \prod_{i=1}^{m} \prod_{j=1}^{n} E_{ij}^{a_{ij}}.$$

The argument will contain m^2 commitments, m^2 ciphertexts and n elements in $\mathbb{Z}_q$, where $N = mn$. Choosing $m = N^{1/3}$ gives a communication complexity of $O(N^{2/3})\kappa$ bits.

When describing the idea, let us first just consider how to get soundness and ignore the issue of zero-knowledge for a moment. In the argument, the prover will prove knowledge of the committed exponents, so let us from now on assume the committed values are well-defined. The prover can compute m^2 ciphertexts

$$D_{i\ell} = \prod_{j=1}^{n} E_{\ell j}^{a_{ij}}.$$

We have $E = E_{pk}(1;R) \prod_{i=1}^m D_{ii} = E_{pk}(1;R) \prod_{i=1}^m \prod_{j=1}^n E_{ij}^{a_{ij}}$. Ignoring R that can be dealt with using standard zero-knowledge techniques all that remains is for the verifier to be convinced $D_{i\ell}$ have been correctly computed. For this purpose the verifier will select challenges $t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$ at random. The prover will open $\prod_{i=1}^m A_i^{t_i}$ to the values $f_1 := \sum_{i=1}^m t_i a_{i1}, \ldots, f_n := \sum_{i=1}^m t_i a_{in}$. The verifier now checks for each $1 \leq \ell \leq m$ that $\prod_{j=1}^n E_{\ell j}^{f_j} = \prod_{i=1}^m D_{i\ell}^{t_i}$. Writing this out we have $\prod_{i=1}^m (\prod_{j=1}^n E_{\ell j}^{a_{ij}})^{t_i} = \prod_{i=1}^m D_{i\ell}^{t_i}$. Since t_i are chosen at random, there is overwhelming probability for one of these checks to fail unless for all i, ℓ we have $D_{i\ell} = \prod_{j=1}^n E_{\ell j}^{a_{ij}}$.

In the argument, we wish to have honest verifier zero-knowledge. We will therefore multiply the $D_{i\ell}$ ciphertexts with random encryptions to avoid leaking information about the exponents. This, however, makes it possible to encrypt anything in $D_{i\ell}$, so to avoid cheating we commit to the plaintexts of those random encryptions and use the commitments to prove that they all cancel out against each other.

Initial message

$a_{01}, \ldots, a_{0n} \leftarrow \mathbb{Z}_q$; $r_0 \leftarrow \mathcal{R}_{ck}$; $A_0 = \mathrm{com}_{ck}(a_{01}, a_{02}, \ldots, a_{0n}; r_0)$
$b_{01}, \ldots, b_{mm} \leftarrow \mathbb{Z}_q$; $r_{01}, \ldots, r_{mm} \leftarrow \mathcal{R}_{ck}$; $b_{mm} := -\sum_{i=1}^{m-1} b_{ii}$; $r_{mm} := -\sum_{i=1}^{m-1} r_{ii}$

$$\begin{array}{lcl} C_{01} := \mathrm{com}_{ck}(b_{01}; r_{01}) & \ldots & C_{0m} := \mathrm{com}_{ck}(b_{0m}; r_{0m}) \\ \vdots & & \vdots \\ C_{m1} := \mathrm{com}_{ck}(b_{m1}; r_{m1}) & \ldots & C_{mm} := \mathrm{com}_{ck}(b_{mm}; r_{mm}) \end{array}$$

$R_{01}, \ldots, R_{mm} \leftarrow \mathcal{R}_{pk}$; $R_{mm} := R - \sum_{i=1}^{m-1} R_{ii}$

$$\begin{array}{llll} D_{01} & := E_{pk}(g^{b_{01}}; R_{01}) \prod_{j=1}^{n} E_{1j}^{a_{0j}} & \cdots\ D_{0m} & := E_{pk}(g^{b_{0m}}; R_{0m}) \prod_{j=1}^{n} E_{mj}^{a_{0j}} \\ \vdots & & & \vdots \\ D_{m1} & := E_{pk}(g^{b_{m1}}; R_{m1}) \prod_{j=1}^{n} E_{1j}^{a_{mj}} & \cdots\ D_{mm} & := E_{pk}(g^{b_{mm}}; R_{mm}) \prod_{j=1}^{n} E_{mj}^{a_{mj}} \end{array}$$

Send $(A_0, C_{01}, \ldots, C_{mm}, D_{01}, \ldots, D_{mm})$ to the verifier

Challenge: $t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$

Answer

For $1 \leq j \leq n$: $f_j := a_{0j} + \sum_{i=1}^{m} t_i a_{ij}$; $z := r_0 + \sum_{i=1}^{m} t_i r_i$

For $1 \leq \ell \leq m$: $F_\ell := b_{0\ell} + \sum_{i=1}^{m} t_i b_{i\ell}$; $z_\ell := r_{0\ell} + \sum_{i=1}^{m} t_i r_{i\ell}$; $Z_\ell := R_{0\ell} + \sum_{i=1}^{m} t_i R_{i\ell}$

Send $(f_1, \ldots, f_n, F_1, \ldots, F_m, z, z_1, \ldots, z_m, Z_1, \ldots, Z_m)$ to the verifier

Verification

Check $A_0 \prod_{i=1}^{m} A_i^{t_i} = \mathrm{com}_{ck}(f_1, \ldots, f_n; z)$

For $1 \leq \ell \leq m$ check

$$C_{0\ell} \prod_{i=1}^{m} C_{i\ell}^{t_i} = \mathrm{com}_{ck}(F_\ell; z_\ell) \qquad \text{and} \qquad E_{pk}(g^{F_\ell}; Z_\ell) \prod_{j=1}^{n} E_{\ell j}^{f_j} = D_{0\ell} \prod_{i=1}^{m} D_{i\ell}^{t_i}$$

Check $\prod_{i=1}^{m} C_{ii} = \mathrm{com}_{ck}(0; 0)$

Check $E = \prod_{i=1}^{m} D_{ii}$

Theorem 3. *The protocol above is a 3-move public coin perfect SHVZK argument of knowledge of* $a_{11}, \ldots, a_{mn}, r_1, \ldots, r_m, R$ *so* $E = E_{pk}(1; R) \prod_{i=1}^{m} \prod_{j=1}^{n} E_{ij}^{a_{ij}}$ *and* $A_i = \mathrm{com}_{ck}(a_{i1}, \ldots, a_{in}; r_i)$.

We refer to the full paper [23] for the proof.

6 Shuffle Argument

Given ciphertexts $\{e_{ij}\}_{i=1,j=1}^{m,n}$ and $\{E_{ij}\}_{i=1,j=1}^{m,n}$ we will give an argument of knowledge of $\pi \in \Sigma_N$ and $\{R_{ij}\}_{i=1,j=1}^{m,n}$ such that for all i, j we have $E_{ij} = e_{\pi^{-1}(ij)} E_{pk}(1; R_{ij})$. The most expensive components of the argument will be a product of committed elements argument and a multi-exponentiation to committed elements argument described in the previous sections. The total size of the argument is therefore $O(m^2 + n)\kappa$ bits, where $N = mn$. With $m = N^{1/3}$ this gives an argument of size $O(N^{2/3})\kappa$ bits.

The argument proceeds in seven steps. First the prover commits to the permutation π, by making a commitment to $1, \ldots, N$ in permuted order. Then the verifier picks challenges $s_1, \ldots, s_m, t_1, \ldots, t_n$ at random. The prover commits to the challenges $s_i t_j$ in permuted order. The prover now proves that she has committed to $s_i t_j$ permuted in the same order as the permutation committed to in the initial commitment. The point of the argument is that since the permutation is committed before seeing the challenges, the prover has no choice in creating the commitment, the random challenges have already been assigned unique slots in the commitment.

The other part of the argument is to use the committed exponentiation technique to show that $\prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j} = E_{pk}(1;R)\prod_{i=1}^{m}\prod_{j=1}^{n} E_{\pi(ij)}^{s_i t_j}$ for some known R. If we look at the plaintext, this implies $\prod_{i=1}^{m}\prod_{j=1}^{n} m_{ij}^{s_i t_j} = \prod_{i=1}^{m}\prod_{j=1}^{n} M_{\pi(ij)}^{s_i t_j}$. With the permutation fixed before the challenges are chosen at random there is overwhelming probability that the argument fails unless for all i,j we have $M_{ij} = m_{\pi^{-1}(ij)}$.

Initial message: The prover sets $a_{\pi(ij)} := m(i-1)+j$. The prover picks $r_{a1},\ldots,r_{am} \leftarrow \mathcal{R}_{ck}$ and sets

$$\begin{aligned} A_1 &:= \mathrm{com}_{ck}(a_{11}, a_{12}, \ldots, a_{1n}; r_{a1}) \\ &\vdots \\ A_m &:= \mathrm{com}_{ck}(a_{m1}, a_{m2}, \ldots, a_{mn}; r_{am}) \end{aligned}$$

First challenge: $s_1,\ldots,s_m,t_1,\ldots,t_n \leftarrow \mathbb{Z}_q$
First answer: We define $b_{\pi(ij)} := s_i t_j$. The prover picks $r_{b1},\ldots,r_{bn} \leftarrow \mathcal{R}_{ck}$ and sets

$$\begin{aligned} B_1 &:= \mathrm{com}_{ck}(b_{11}, b_{12}, \ldots, b_{1n}; r_{b1}) \\ &\vdots \\ B_m &:= \mathrm{com}_{ck}(b_{m1}, b_{m2}, \ldots, b_{mn}; r_{bm}) \end{aligned}$$

Second challenge: $\lambda \leftarrow \mathbb{Z}_q$
Answer: Make a 4-move argument of knowledge of $\pi \in \Sigma_N$ and openings of $A_1^{\lambda}B_1,\ldots,A_m^{\lambda}B_m$ so they contain a permutation of the N values $\lambda(m(i-1)+j)+s_i t_j$. Observe, the first move of this argument can be made in parallel with the second challenge so we only use three additional moves.
Make a 3-move argument of knowledge of b_{ij}, r_{bi}, R so

$$\begin{aligned} B_1 &= \mathrm{com}_{ck}(b_{11}, b_{12}, \ldots, b_{1n}; r_{b1}) \\ &\vdots \\ B_m &= \mathrm{com}_{ck}(b_{m1}, b_{m2}, \ldots, b_{mn}; r_{bm}) \end{aligned}$$

$$\text{and} \qquad \prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j} = E_{pk}(1;R)\prod_{i=1}^{m}\prod_{j=1}^{n} E_{ij}^{b_{ij}}.$$

Theorem 4. *The protocol is a 7-move public coin perfect SHVZK argument of knowledge of $\pi \in \Sigma$ and $R_{ij} \in \mathcal{R}_{pk}$ so $E_{ij} = e_{\pi^{-1}(ij)}E_{pk}(1;R_{ij})$.*

We refer to the full paper [23] for the proof.

7 Efficient Verification

The small size of the argument gives a corresponding low cost of verification. There are, however, $2N$ ciphertexts that we must exponentiate in the verification. In this section we show that the verifier computation can be reduced to making multi-exponentiations of the ciphertexts to small exponents.

7.1 Prover-Assisted Multi-exponentiation

In our shuffle argument, the verifier has to compute

$$\prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j}.$$

The prover can assist this computation by computing $D_1, \ldots, D_n$ as $D_j := \prod_{i=1}^{m} e_{ij}^{s_i}$. The verifier can then compute

$$\prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j} = \prod_{j=1}^{m} D_j^{t_j}.$$

What remains is for the verifier to check that the ciphertexts are correct, which can be done by verifying

$$\prod_{j=1}^{n} D_j^{\alpha_j} = \prod_{i=1}^{m}(\prod_{j=1}^{n} e_{ij}^{\alpha_j})^{s_i}$$

for randomly chosen α_j. Since the check is done off-line, the verifier can use small exponents α_j, say, 32-bit exponents. This trick reduces the amount of verifier computation that is needed for computing $\prod_{i=1}^{m}\prod_{i=1}^{n} e_{ij}^{s_i t_j}$ to one m-exponentiation to exponents from $\mathbb{Z}_q$ and $m+1$ n-exponentiations to small exponents.

When m is small, this strategy may actually end up increasing the communication complexity of the shuffle. However, the exact same method can be employed when we let the verifier compute the t_j-values as products the n products of $\psi_1, \ldots, \psi_{n_1}$ and $\tau_1, \ldots, \tau_{n_2}$ where $n = n_1 n_2$. If we choose $n_2 = \sqrt{N}$ for instance, we get that the prover only sends $\sqrt{N}$ ciphertexts to the verifier. The verifier then makes $\sqrt{N}$-multi-exponentiations to small exponents $\alpha_1, \ldots, \alpha_{\sqrt{N}}$.

7.2 Randomized Verification

In the argument for multi-exponentiation to committed exponents, the verifier must check m equalities of the form

$$E_{pk}(g^{F_\ell}; Z_\ell)\prod_{j=1}^{n} E_{\ell j}^{f_j} = D_{0\ell}\prod_{i=1}^{m} D_{i\ell}^{t_i}.$$

This can be done off-line in a randomized way by picking $\alpha_1, \ldots, \alpha_m$ at random and testing whether

$$E_{pk}(g^{\sum_{\ell=1}^{m}\alpha_\ell F_\ell}; \sum_{\ell=1}^{m}\alpha_\ell Z_\ell)\prod_{j=1}^{n}\left(\prod_{\ell=1}^{m} E_{\ell j}^{\alpha_\ell}\right)^{f_j} = \prod_{\ell=1}^{m}\left(E_{pk}(g^{F_\ell}; Z_\ell)\prod_{j=1}^{n} E_{\ell j}^{f_j}\right)^{\alpha_\ell}$$
$$= \prod_{\ell=1}^{m} D_{0\ell}^{\alpha_\ell}\prod_{i=1}^{m}\left(\prod_{\ell=1}^{m} D_{i\ell}^{\alpha_\ell}\right)^{t_i}.$$

This way, we make n m-multi-exponentiations to small exponents α_ℓ and one n-multi-exponentiation to larger exponents f_j.

8 Comparison

Let us compare our shuffle argument with the most efficient arguments for correctness of a shuffle of ElGamal ciphertexts in the literature. Furukawa and Sako [17] suggested an efficient argument for correctness of a shuffle based on committing to a permutation matrix. This scheme was further refined by Furukawa [15]. We will use Groth and Lu's [24] estimates for the complexity of Furukawa's scheme. Neff [30,31] gave an efficient interactive proof for correctness of a shuffle. Building on those ideas Groth [21] suggested a perfect SHVZK argument for correctness of a shuffle. Our shuffle argument builds on Neff's and Groth's schemes.

We will compare the schemes using an elliptic curve of prime order q. We use $|q| = 256$ so SHA256 can be used to choose the public coin challenges. We measure the communication complexity in bits and measure the prover and verifier computation in single exponentiations. By this we mean that in all schemes, we count the cost of a multi-exponentiation to n exponents as n single exponentiations. We compare the most efficient shuffle arguments in Table 1. Section 7 offer a couple of speedup techniques.

Table 1. Comparison of shuffle arguments for $N = mn$ ElGamal ciphertexts

Elliptic curve Group order: $\|q\| = 256$	Furukawa-Sako [17]	Groth [21]	Furukawa [15,24]	proposed
Prover (single expo.)	$8N$	$6N$	$7N$	$3mN + 5N$
Verifier (single expo.)	$10N$	$6N$	$8N$	$4N + 3n$
Prover's communication (bits)	$1280N$	$768N$	$768N$	$768m^2 + 768n$
Rounds	3	7	3	7

If we employ the randomization techniques from Section 7 then the prover's cost increases by $2N$ exponentiations, whereas the verifier's complexity reduces to $4N$ small exponentiations and $m^2 + 3n$ exponentiations to full size exponents from $\mathbb{Z}_q$.

For all schemes it holds that multi-exponentiation techniques can reduce their cost, see e.g. Lim [27]. We refer to the full paper of Groth [21] for a discussion of randomization techniques and other tricks that can be used to reduce the computational complexity of all the shuffle arguments. An additional improvement of our scheme is to let the prover assist the verifier in computing the multi-exponentiation $\prod_{i=1}^{m} \prod_{j=1}^{n} e_{ij}^{s_i t_j}$, see Section 7. Table 2 has back-of-the-envelope estimates when we compare an optimized version of our scheme to that of Groth [21]. We assume that we are shuffling $N = 100,000$ ElGamal ciphertexts with parameters $m = 10, n = 10,000$ so $N = mn$.

Table 2. Comparison of shuffle arguments for $100,000$ ElGamal ciphertexts

	Groth [21]	proposed
Prover's computation	$18 \cdot 10^6$ mults (18 sec.)	$143 \cdot 10^6$ mults (143 sec.)
Verifier's computation	$14 \cdot 10^6$ mults (14 sec.)	$5 \cdot 10^6$ mults (5 sec.)
Prover's communication	77 Mbits	8 Mbits

We count the computational cost in the number of multiplications. In parenthesis we are giving timing estimates assuming the use of equipment where a multiplication takes $1\mu s$, which is conservative given today's equipment. We only count the cost of the shuffle argument in Table 2, not the cost of computing the shuffle or the size of the shuffle (51 Mbits).

References

1. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 437–447. Springer, Heidelberg (1998)
2. Abe, M., Hoshino, F.: Remarks on mix-network based on permutation networks. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 317–324. Springer, Heidelberg (2001)
3. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM 45(3), 501–555 (1998)
4. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. Journal of the ACM 45(1), 70–122 (1998)
5. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS, pp. 62–73 (1993)
7. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Short PCPs verifiable in polylogarithmic time. In: IEEE Conference on Computational Complexity, pp. 120–134 (2005)
8. Brassard, G., Chaum, D., Crèpeau, C.: Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences 37(2), 156–189 (1988)
9. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
10. Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000)
11. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
12. Dinur, I.: The PCP theorem by gap amplification. Journal of the ACM 54(3) (2007)
13. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
14. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1986)
15. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 88-A(1), 172–188 (2005)
16. Furukawa, J., Miyauchi, H., Mori, K., Obana, S., Sako, K.: An implementation of a universally verifiable electronic voting scheme based on shuffling. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 16–30. Springer, Heidelberg (2002)
17. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
18. Garay, J.A., MacKenzie, P.D., Yang, K.: Strengthening zero-knowledge protocols using signatures. Journal of Cryptology 19(2), 169–209 (2006)

19. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: FOCS, pp. 102–113 (2003), http://eprint.iacr.org/2003/034
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proofs. SIAM Journal of Computing 18(1), 186–208 (1989)
21. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002), http://eprint.iacr.org/2005/246
22. Groth, J.: Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS, PhD thesis. pp. xii+119 (2004)
23. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle (2008), http://www.brics.dk/~jg/PCPShuffle.pdf
24. Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 377–392. Springer, Heidelberg (2007)
25. Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient arguments without short PCPs. In: CCC, pp. 278–291 (2007)
26. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: STOC, pp. 723–732 (1992)
27. Lim, C.H.: Efficient multi-exponentiation and application to batch verification of digital signatures (2000), http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps
28. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. Journal of Cryptology 16(3), 143–184 (2003)
29. Micali, S.: Computationally sound proofs. SIAM Journal of Computing 30(4), 1253–1298 (2000)
30. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM CCS, pp. 116–125 (2001)
31. Neff, C.A.: Verifiable mixing (shuffling) of ElGamal pairs (2003), http://www.votehere.net/vhti/documentation/egshuf.pdf
32. Nguyen, L., Safavi-Naini, R., Kurosawa, K.: A provably secure and effcient verifiable shuffle based on a variant of the Paillier cryptosystem. Journal of Universal Computer Science 11(6), 986–1010 (2005)
33. Nguyen, L., Safavi-Naini, R., Kurosawa, K.: Verifiable shuffles: a formal model and a Paillier-based three-round construction with provable security. International Journal of Information Security 5(4), 241–255 (2006)
34. Onodera, T., Tanaka, K.: Shufle for Paillier's encryption scheme. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E88-A(5), 1241–1248 (2005)
35. Paillier, P.: Public-key cryptosystems based on composite residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–239. Springer, Heidelberg (1999)
36. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
37. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
38. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)

Precise Concurrent Zero Knowledge

Omkant Pandey[1], Rafael Pass[2,⋆], Amit Sahai[1,⋆⋆], Wei-Lung Dustin Tseng[2,⋆⋆⋆], and Muthuramakrishnan Venkitasubramaniam[2]

[1] University of California Los Angeles, California
{omkant,sahai}@cs.ucla.edu
[2] Cornell University, New York
{rafael,wdtseng,vmuthu}@cs.cornell.edu

Abstract. *Precise zero knowledge* introduced by Micali and Pass (STOC'06) guarantees that the view of any verifier V can be simulated in time closely related to the *actual* (as opposed to worst-case) time spent by V in the generated view. We provide the first constructions of precise concurrent zero-knowledge protocols. Our constructions have essentially optimal precision; consequently this improves also upon the previously tightest non-precise concurrent zero-knowledge protocols by Kilian and Petrank (STOC'01) and Prabhakaran, Rosen and Sahai (FOCS'02) whose simulators have a quadratic worst-case overhead. Additionally, we achieve a statistically-precise concurrent zero-knowledge property—which requires simulation of unbounded verifiers participating in an unbounded number of concurrent executions; as such we obtain the first (even non-precise) concurrent zero-knowledge protocols which handle verifiers participating in a super-polynomial number of concurrent executions.

1 Introduction

Zero-knowledge interactive proofs, introduced by Goldwasser, Micali and Rackoff [GMR85] are constructs allowing one player (called the Prover) to convince another player (called the Verifier) of the validity of a mathematical statement $x \in L$, while providing no additional knowledge to the Verifier. The zero-knowledge property is formalized by requiring that the view of any PPT verifier V in an interaction with a prover can be "indistinguishably reconstructed" by

⋆ This material is based upon work supported under a I3P Identity Management and Privacy Grant.

⋆⋆ This research was supported in part by NSF ITR and Cybertrust programs (including grants 0627781, 0456717, 0716389, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Okawa Research Award, and an Alfred P. Sloan Foundation Research Fellowship. First author was supported in part from third author's grants.

⋆⋆⋆ This material is based upon work supported under a National Science Foundation Graduate Research Fellowship, and a NSERC Canada Julie-Payette Fellowship.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 397–414, 2008.

a PPT simulator S, interacting with no one, on input just x. Since whatever V "sees" in the interaction can be reconstructed by the simulator, the interaction does not yield anything to V that cannot already be computed with just the input x. Because the simulator is allowed to be an arbitrary PPT machine, this traditional notion of ZK only guarantees that the *class* of PPT verifiers learn nothing. To measure the knowledge gained by a particular verifier, Goldreich, Micali and Wigderson [GMW87] (see also [Gol01]) put forward the notion of *knowledge tightness*: intuitively, the "tightness" of a simulation is a function relating the (worst-case) running-time of the verifier and the (expected) running-time of the simulator—thus, in a knowledge-tight ZK proof, the verifier is guaranteed not to gain more knowledge than what it could have computed in time closely related to its *worst-case* running-time.

Micali and Pass [MP06] recently introduced the notion of *precise zero knowledge* (originally called *local* ZK in [MP06]). In contrast to traditional ZK (and also knowledge-tight ZK), precise ZK considers the knowledge of an individual verifier in an *individual execution*—it requires that the view of any verifier V, in which V takes t computational steps, can be reconstructed in time closely related to t—say $2t$ steps. More generally, we say that a zero-knowledge proof has precision $p(\cdot,\cdot)$ if the simulator uses at most $p(n,t)$ steps to output a view in which V takes t steps on common input an instance $x \in \{0,1\}^n$.

This notion thus guarantees that the verifier does not learn more than what can be computed in time closely related to the *actual* time it spent in an interaction with the prover. Such a guarantee is important, for instance, when considering knowledge of "semi-easy" properties of the instance x, considering proofs for "semi-easy" languages L, or when considering *deniability* of interactive protocols (see [MP06,Pas06] for more discussion).

The notion of precise ZK, however, only considers verifiers in a stand-alone execution. A more realistic model introduced by Dwork, Naor and Sahai [DNS98], instead considers the execution of zero-knowledge proofs in an asynchronous and concurrent setting. More precisely, we consider a single adversary mounting a coordinated attack by acting as a verifier in many concurrent *sessions* of the same protocol. Concurrent zero-knowledge proofs are significantly harder to construct and analyze.

Richardson and Kilian [RK99] constructed the first concurrent zero-knowledge argument in the standard model (without any extra set-up assumptions). Their protocol requires $O(n^\epsilon)$ number of rounds. Kilian and Petrank [KP01] later improved the round complexity to $\tilde{O}(\log^2 n)$. Finally, Prabhakaran, Rosen and Sahai [PRS02] provided a tighter analyis of the [KP01] simulator showing that $\tilde{O}(\log n)$ rounds are sufficient. However, none of the simulators exhibited for these protocols are precise, leaving open the following question:

Do there exist precise concurrent zero-knowledge proofs (or arguments)?

In fact, the simulators of [RK99,KP01,PRS02] are not only imprecise, but even the overhead of the simulator with respect to the *worst-case* running-time of the verifier—as in the definition of knowledge tightness—is high. The simulator of [RK99] had *worst-case precision* $p(n,t) = t^{O(\log_n t)}$—namely, the running-time of their simulator for a verifier V with *worst-case* running-time t is $p(n,t)$ on input a statement $x \in \{0,1\}^n$. This was significantely improved by [KP01] who obtained a quadratic worst-case precision, namely $p(n,t) = O(t^2)$; the later result by [PRS02] did not improve upon this, leaving open the following question:

> *Do there exist concurrent zero-knowledge arguments (or proofs) with sub-quadratic worst-case precision?*

Our Results. Our main result answers both of the above questions in the affirmative. In fact, we present concurrent zero-knowledge protocols with essentially optimal precision. Our main lemma shows the following.

Lemma 1 (Main Lemma). *Assuming the existence of one-way functions, for every $k, g \in \mathbb{N}$ such that $k/g \in \omega(\log n)$, there exists an $O(k)$-round concurrent zero knowledge argument with precision $p(t) \in O(t \cdot 2^{\log_g t})$ for all languages in $\mathcal{NP}$.*

By setting k and g appropriately, we obtain a simulation with near-optimal precision.

Theorem 1. *Assuming the existence of one-way functions, for every $\varepsilon > 0$, there exists a $\omega(\log n)$-round concurrent zero knowledge argument for all languages in $\mathcal{NP}$ with precision $p(t) = O(t^{1+\varepsilon})$.*

Theorem 2. *Assuming the existence of one-way functions, for every $\varepsilon > 0$, there exists an $O(n^\varepsilon)$-round concurrent zero knowledge argument for all languages in $\mathcal{NP}$ with precision $p(t) = O(t 2^{\frac{2}{\epsilon} \log_n t})$. As a corollary, we obtain the following: For every $\epsilon > 0$, there exists an $O(n^\varepsilon)$-round protocol $\langle P, V \rangle$ such that for every $c > 0$, $\langle P, V \rangle$ is a concurrent zero knowledge argument with precision $p(n,t) = O(t)$ with respect to verifiers with running time bounded by n^c for all languages in $\mathcal{NP}$.*

Finally, we also construct statistically-precise concurrent ZK arguments for all of $\mathcal{NP}$, which requires simulation of *all* verifiers, even those having a priori unbounded running time.

Theorem 3. *Assume the existence of* claw-free permutations*, then there exists a poly(n)-round statistically precise concurrent zero-knowledge argument for all of $\mathcal{NP}$ with precision $p(t) = t^{1+\frac{1}{\log n}}$.*

As far as we know, this is the first (even non-precise) concurrent ZK protocol which handles verifiers participating in an unbounded number of executions. Previous work on statistical concurrent ZK also considers verifiers with an unbounded running-time; however, those simulations break down if the verifier can participate in a super-polynomial number of executions.

Our Techniques. Micali and Pass show that only trivial languages have black-box simulator with polynomial precision [MP06]. To obtain precise simulation, they instead "time" the verifier and then try to "cut off" the verifier whenever it attempts to run for too long. A first approach would be to adapt this technique to the simulators of [RK99,KP01,PRS02]. However, a direct application of this cut-off technique breaks down the correctness proof of these simulators.

To circumvent this problem, we instead introduce a new simulation technique, which rewinds the verifier *obliviously* based on *time*. In a sense, our simulator is not only oblivious of the content of the messages sent by the verifier (as the simulator by [KP01]), but also oblivious to when messages are sent by the verifier!

The way our simulator performs rewindings relies on the rewinding schedule of [KP01], and our analysis relies on that of [PRS02]. However, obtaining our results requires us to both modify and generalize this rewinding schedule and therefore also change the analysis. In fact, we cannot use the same rewinding schedule as KP/PRS as this yields at best a quadratic *worst-case* precision.

2 Definitions and Preliminaries

Notation. Let L denote an NP language and R_L the corresponding NP-relation. Let $(\mathcal{P},\mathcal{V})$ denote an interactive proof (argument) system where $\mathcal{P}$ and $\mathcal{V}$ are the prover and verifier algorithms respectively. By $\mathcal{V}^*(x, z, \bullet)$ we denote a non-uniform *concurrent adversarial* verifier with common input x and auxiliary input (or advice) z whose random coins are fixed to a sufficiently long string chosen uniformly at random; $\mathcal{P}(x, w, \bullet)$ is defined analogously where $w \in R_L(x)$.

Note that $\mathcal{V}^*$ is a *concurrent adversarial* verifier. Formally, it means the following. Adversary $\mathcal{V}^*$, given an input $x \in L$, interacts with an unbounded number of independent copies of $\mathcal{P}$ (all on common input x)[1]. An execution of a protocol between a copy of $\mathcal{P}$ and $\mathcal{V}^*$ is called a *session*. Adversary $\mathcal{V}^*$ can interact with all the copies at the *same* time (i.e., concurrently), interleaving messages from various sessions in any order it wants. That is, $\mathcal{V}^*$ has control over the scheduling of messages from various sessions. In order to implement a scheduling, $\mathcal{V}^*$ concatenates each message with the session number to which the next scheduled message belongs. The prover copy corresponding to that session then immediately replies to the verifier message as specified by the protocol. The *view* of concurrent adversary $\mathcal{V}^*$ in a concurrent execution consists of the common input x, the sequence of prover and verifier messages exchanged during the interaction, and the contents of the random tape of $\mathcal{V}^*$.

Let $\text{VIEW}_{\mathcal{V}\ (x,z,\bullet)}$ be the random variable denoting the view of $\mathcal{V}^*(x, z, \bullet)$ in a *concurrent* interaction with the copies of $\mathcal{P}(x, w, \bullet)$. Let $\text{VIEW}_{\mathcal{S}_{\mathcal{V}}\ (x,z,\bullet)}$ denote the view output by the simulator. When the simulator's random tape is *fixed* to

[1] We remark that instead of a single fixed theorem x, $\mathcal{V}^*$ can be allowed to adaptively choose provers with different theorems x'. For ease of notation, we choose a single theorem x for all copies of $\mathcal{P}$. This is not actually a restriction and our results hold even when $\mathcal{V}^*$ adaptively chooses different theorems.

r, its output is instead denoted by $\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}\ _{(x,z,r)}$. Finally, let $T_{\mathcal{S}_{\mathcal{V}}}\ _{(x,z,r)}$ denote the *steps* taken by the simulator and let $T_{\mathcal{V}}$ (VIEW) denote the steps taken by $\mathcal{V}^*$ in the view VIEW. For ease of notation, we will use $\text{VIEW}_{\mathcal{V}}$ to abbreviate $\text{VIEW}_{\mathcal{V}}\ _{(x,z,\bullet)}$, and $\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}$ to abbreviate $\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}\ _{(x,z,\bullet)}$, whenever it is clear from the context.

Definition 1 (Precise Concurrent Zero Knowledge). *Let $p : N \times N \rightarrow N$ be a monotonically increasing function. $(\mathcal{P},\mathcal{V})$ is a* concurrent zero knowledge *proof (argument) system with* precision p *if for every non-uniform* probabilistic polynomial time $\mathcal{V}^*$*, the following conditions hold:*

1. *For all $x \in L$, $z \in \{0,1\}^*$, the following distributions are computationally indistinguishable over L:*
   $\left\{\text{VIEW}_{\mathcal{V}}\ _{(x,z,\bullet)}\right\}$ *and* $\left\{\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}\ _{(x,z,\bullet)}\right\}$
2. *For all $x \in L$, $z \in \{0,1\}^*$, and every sufficiently long $r \in \{0,1\}^*$, it holds that:*
   $T_{\mathcal{S}_{\mathcal{V}}}\ _{(x,z,r)} \leq p(|x|, T_{\mathcal{V}}\ (\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}\ _{(x,z,r)}))$.

When there is no restriction on the running time of $\mathcal{V}^*$ and the first condition requires the two distributions to be statistically close (resp., identical), we say $(\mathcal{P}, \mathcal{V})$ is statistical (resp., perfect) zero knowledge.

Next, we briefly describe some of the cryptographic tools used in our construction.

Special Honest Verifier Zero Knowledge (HVZK). A (three round) protocol is special-HVZK if, given the verifier's challenge in advance, the simulator can construct the first and the last message of the protocol such that the simulated view is computationally indistinguishable from the real view of an honest verifier. The Blum-Hamiltonicity protocol [Blu87] used in our construction is special-HVZK. When the simulated view is identical to the real view, we say the protocol is perfect-special-HVZK.

View Simulation. We assume familarity with the notion of "simulating the verifier's view". In particular, one can fix the random tape of the adversarial verifier $\mathcal{V}^*$ during simulation, and treat $\mathcal{V}^*$ as a deterministic machine.

Perfectly/Statistically Binding Commitments. We assume familiarity with "perfectly/statistically binding and computationally hiding" commitment schemes. Such commitment schemes are known based on the existence of one way function [Nao91,HILL99]. Naor's scheme has a two round commit phase where the first message is sent by the receiver. Thereafter, the sender can create the commitment using a randomized algorithm, denoted $c \leftarrow \text{COM}_{\text{PB}}(v)$. The decommitment phase is only one round, in which the sender simply sends v and the randomness used, to the receiver. This will be denoted by $(v, r) \leftarrow \text{DCOM}_{\text{PB}}(c)$. More on commitment schemes appears in the full version of this paper [PPS+07].

3 Our Protocol

We describe our Precise Concurrent Zero-Knowledge Argument, PCZK, in Figure 1. It is a slight variant of the PRS-protocol [PRS02]; in fact, the only difference is that we pad each verifier message with the string 0^l if our zero knowledge simulator (found in Figure 5) requires l steps of computation to produce the next message (l grows with the size of x). For simplicity, we use perfectly binding commitments in PCZK, although it suffices to use statistically binding commitments, which in turn rely on the existence of one way functions. The parameter k determines the round complexity of PCZK.

PCZK(k): A Protocol for Precise Concurrent Zero Knowledge Arguments.
All verifier messages are padded with the string 0^l where l is the running time required by our simulator (Figure 5) to compute the next prover message.

(Stage 1)

P0: Select the first message of the *perfectly binding* commitment scheme $(\text{COM}_{\text{PB}}, \text{DCOM}_{\text{PB}})$ and send it to $\mathcal{V}$.

V0: Select $\sigma \overset{R}{\leftarrow} \{0,1\}^n$ and set $\beta \leftarrow \text{COM}_{\text{PB}}(\sigma)$. Now select strings $\sigma^0_{i,j} \overset{R}{\leftarrow} \{0,1\}^n$ and $\sigma^1_{i,j} \leftarrow \{0,1\}^n$ such that $\sigma^0_{i,j} \oplus \sigma^1_{i,j} = \sigma$, for $i,j = 1,2,\ldots,k$ (total $2k^2$ strings). Create commitments $\beta^b_{i,j} \leftarrow \text{COM}_{\text{PB}}(\sigma^b_{i,j})$ for all values of i,j and $b = 0,1$. Send $\beta, \{\beta^0_{i,j}\}^k_{i,j=1}, \{\beta^1_{i,j}\}^k_{i,j=1}$, to $\mathcal{P}$.

For $j = 1,2,\ldots,k$ proceed as follows.

Pj: Select $r_j \in \{0,1\}^k$ uniformly at random and send it to the verifier.

Vj: Let $r_{i,j}$ denote the i^{th} bit of r_j. Then, send $\sigma^0_{i,j}$ (resp., $\sigma^1_{i,j}$) and the decommitment information of $\beta^0_{i,j}$ (resp., $\beta^1_{i,j}$), if $r_{i,j} = 0$ (resp., if $r_{i,j} = 1$) to the prover.

(Stage 2)

p1: If $\mathcal{V}$ failed to properly decommit in Step Vj, for *any* $j \in [k]$, abort the protocol. Otherwise, run n parallel and *independent* copies of BH-prover (Figure 2) and send the n prover messages $\widehat{\text{p1}}$ to the verifier.

v1: Reveal the challenge σ and send decommitment information for *all* commitments which are unopened so far. Each bit of σ can be thought of as verifier's challenge in Step $\widehat{\text{v1}}$ of BH-protocol.

p2: Prover verifies that all the decommitments are proper and that $\sigma = \sigma^0_{i,j} \oplus \sigma^1_{i,j}$. If yes, execute the step $\widehat{\text{p2}}$ for each of the n parallel copies of the BH-protocol.

v2: Verify each of the n parallel proofs as described in $\widehat{\text{v2}}$. If all n $\widehat{\text{v2}}$ steps are accepting, accept the proof, otherwise reject the proof.

Fig. 1. Our Precise Concurrent Zero Knowledge Protocol

Since our PCZK-protocol is just an instantiation of the PRS-protocol (with extra padding), it is both complete and sound.

The Blum-Hamiltonicity(BH) Protocol [Blu87].

$\widehat{\text{p1}}$: Choose a random permutation π of vertices V. Commit to the adjacency matrix of the permuted graph, denoted $\pi(G)$, and the permutation π, using a *perfectly binding* commitment scheme. Notice that the adjacency matrix of the permuted graph contains a 1 in position $(\pi(i), \pi(j))$ if $(i, j) \in E$. Send both the commitments to the verifier.

$\widehat{\text{v1}}$: Select a bit $\sigma \in \{0, 1\}$, called the *challenge*, uniformly at random and send it to the prover.

$\widehat{\text{p2}}$: If $\sigma = 0$, send π along with the decommitment information of *all* commitments. If $\sigma = 1$ (or anything else), decommit all entries $(\pi(i), \pi(j))$ with $(i, j) \in C$ by sending the decommitment information for the corresponding commitments.

$\widehat{\text{v2}}$: If $\sigma = 0$, verify that the revealed graph is identical to the graph $\pi(G)$ obtained by applying the revealed permutation π to the common input G. If $\sigma = 1$, verify that all the revealed values are 1 and that they form a cycle of length n. In both cases, verify that all the revealed commitments are correct using the decommitment information received. If the corresponding conditions are satisfied, accept the proof, otherwise reject the proof.

Fig. 2. The Blum-Hamiltonicity protocol used in PCZK

4 Our Simulator and Its Analysis

4.1 Overview

At a high level, our simulator receives several opportunities to rewind the verifier and extract the "trapdoor" σ that will allow it to complete the simulation. More precisely, our simulator will attempt to rewind the verifier in one of the k "slots" (i.e. a message pair $\langle(\text{Pj}), (\text{Vj})\rangle$) in the first stage. If at any point it obtains the decommitment information for two different challenges (Pj), the simulator can extract the secret σ (that the verifier sends in the Stage 2) and simulate the rest of the protocol using the special-HVZK property of the BH-protocol.

To handle concurrency and precision, consider first the KP/PRS simulator. This simulator relies on a static and *oblivious* rewinding schedule, where the simulator rewinds the verifier after some fixed number of messages, independent of the message content. Specifically, the total number of verifier messages over all sessions are divided into two halves. The KP/PRS-rewinding schedule recursively invokes itself on each of the halves twice (completing two runs of the first half before proceeding to the two runs of the second half). The first run of each half is called the *primary* thread, and the latter is called the *secondary* thread. As shown in [KP01,PRS02], after the verifier commits to σ in any given session s, the KP/PRS-simulator gets several opportunities to extract it *before* Stage 2 of session s begins. We also call the thread of execution in the final output by the simulator the *main thread*. The KP/PRS-simulator keeps uses the secondary threads (recursively) as the main thread; all other threads, used to gather useful information for extracting σ, are called *look-ahead* threads. However, since the verifier's running time in look-ahead threads could be significantly longer than its running time in the main thread, the KP/PRS-simulator is not precise.

On the other hand, consider the precise simulation by Micali and Pass [MP06]. When rewinding a verifier, the MP simulator cuts off the second run of the verifier if it takes more time than the first run, and outputs the view of the verifier on the first run. Consequently, the running time of the simulator is proportional to the running time of the verifier on the output view. In order to apply the MP "cut" strategy on top of the KP/PRS-simulator, we need to use the primary thread (recursively) as the main output thread, and "cut" the secondary thread with respect to the primary thread. However, this cut-off will cause the simulator to abort more often, which significantly complicates the analysis.

To circumvent the above problems, we introduce a new simulation technique. For simplicity, we first present a simulator that knows an upper bound to the running-time of the verifier. Later, using a standard "doubling" argument, we remove this assumption. Like the KP/PRS-rewinding schedule, our simulator is oblivious of the verifier. But instead of rewinding based on the number of messages sent, we instead rewind based on the number of *steps* taken by the verifier (and thus this simulator is oblivious not only to the content of the messages sent by the verifier, but also to the time when these messages are sent!). In more detail, our simulator divides the total *running time* T of $\mathcal{V}^*$ into two halves and executes itself recursively on each half twice. In each half, we execute the primary and secondary threads in *parallel*. As we show later, this approach results in a simulation with quadratic precision.

To improve the precision, we further generalize the KP/PRS rewinding schedule. Instead of dividing T into *two* halves, we instead consider a simulator that divides T into g parts, where g is called the *splitting factor*. By choosing g appropriately, we are able to provide precision $p(t) \in O(t^{1+\epsilon})$ for every constant ϵ. Furthermore, we show how to achieve essentially *linear* precision by adjusting both k (the round complexity of our protocol) and g appropriately.

4.2 Worst Case Quadratic Simulation

We first describe a procedure that takes as input a parameter t and simulates the view of the verifier for t steps. The SIMULATE procedure described in Figure 3 employs the KP rewinding method with the changes discussed earlier. In Stage 1, SIMULATE simply generates uniformly random messages. SIMULATE attempts to extract σ using rewindings, and uses the special honest-verifier ZK property of the BH protocol to generate Stage 2 messages. If the extraction of σ fails, it outputs $\perp$. The parameter st is the state of $\mathcal{V}^*$ from which the simulation should start, and the parameter $\mathcal{H}$ is simply a global history of all "useful messages" for extracting σ.[2]

Let st_0 be the initial state of $\mathcal{V}^*$ and $d = d_t$ be the maximum recursion depth of SIMULATE$(t, \mathsf{st}_0, \emptyset)$. The actual precise simulator constructed in the next section uses SIMULATE as a sub-routine, for which we show some properties below. In

[2] For a careful treatment of $\mathcal{H}$, see [Ros04].

Proposition 2, we show that SIMULATE$(t, \mathsf{st}_0, \emptyset)$ has a worst case running time of $O(t^2)$, and in Proposition 3 we show that SIMULATE outputs $\perp$ with negligible probability.

The SIMULATE$(t, \mathsf{st}, \mathcal{H})$ Procedure.

1. If $t = 1$,
   (a) If the next scheduled message, p_u, is a first stage prover message, choose p_u uniformly. Otherwise, if p_u is a second stage prover message, compute p_u using the PROVE procedure (Figure 4). Feed p_u to the verifier. If the next scheduled message is verifier's message, run the verifier from its current state st for exactly 1 step. If an output is received then set $\mathrm{v}_u \leftarrow \mathcal{V}^*(\mathsf{hist}, \mathrm{p}_u)$. Further, if v_u is a first stage verifier message, store v_u in $\mathcal{H}$.
   (b) Update st to the current state of $\mathcal{V}^*$. Output $(\mathsf{st}, \mathcal{H})$.
2. Otherwise (i.e., $t > 1$),
   (a) Execute the following two processes in *parallel*:
      i. $(\mathsf{st}_1, \mathcal{H}_1) \leftarrow$ SIMULATE$(t/2, \mathsf{st}, \mathcal{H})$. (primary process)
      ii. $(\mathsf{st}_2, \mathcal{H}_2) \leftarrow$ SIMULATE$(t/2, \mathsf{st}, \mathcal{H})$. (secondary process)

      Merge $\mathcal{H}_1$ and $\mathcal{H}_2$. Set the resulting table equal to $\mathcal{H}$.
   (b) Next, execute the following two processes in *parallel*, starting from st_1,
      i. $(\mathsf{st}_3, \mathcal{H}_3) \leftarrow$ SIMULATE$(t/2, \mathsf{st}_1, \mathcal{H})$. (primary process)
      ii. $(\mathsf{st}_4, \mathcal{H}_4) \leftarrow$ SIMULATE$(t/2, \mathsf{st}_1, \mathcal{H})$[3]. (secondary process)
   (c) Merge $\mathcal{H}_3$ and $\mathcal{H}_4$. Set the resulting table equal to $\mathcal{H}$.
      Output $(\mathsf{st}_3, \mathcal{H})$ and the view of $\mathcal{V}^*$ on the thread connecting st, st_1, and st_3.

Fig. 3. The time-based oblivious simulator

Proposition 2 (Running Time of simulate). SIMULATE$(t, \cdot, \cdot)$ has worst-case running time $O(t^2)$.

Proof. We partition the running time of SIMULATE into the time spent emulating $\mathcal{V}^*$, and the time spent simulating the prover (i.e. generating prover messages). By construction, SIMULATE$(t, \cdot, \cdot)$ spends time at most t emulating $\mathcal{V}^*$ on main thread. Furthermore, the number of parallel executions double per level of recursion. Thus, the time spent in simulating $\mathcal{V}^*$ by SIMULATE$(t, \cdot, \cdot)$ is $t \cdot 2^d$, where the d is the maximum depth of recursion. Since $d = d_t = \lceil \log_2 t \rceil \leq 1 + \log_2 t$, we conclude that SIMULATE spends at most $2t^2$ steps emulating $\mathcal{V}^*$. To compute the time spent simulating the prover, recall that the verifier pads each messages with 0^l if the SIMULATE requires l steps of computation to generate the next message. Therefore, SIMULATE always spends less time simulating the prover than $\mathcal{V}^*$ giving us a bound of $2 \cdot 2t^2 = 4t^2$ on the total running time. □

[3] In the case where t does not divide evenly into two, we use $\lfloor t/2 \rfloor + 1$ in step (2a), and $\lfloor t/2 \rfloor$ in step (2b).

The PROVE Procedure.

Let $s \in [m]$ be the session for which the prove procedure is invoked. The procedure outputs either p1 or p2, whichever is required by $\mathcal{S}_{\mathcal{V}}$. Let hist denote the set of messages exchanged between $\mathcal{S}_{\mathcal{V}}$ and $\mathcal{V}^*$ in the *current* thread. The PROVE procedure works as follows.

1. If the verifier has aborted in any of the k first stage messages of session s (i.e., hist contains Vj=ABORT for $j \in [k]$ of session s), abort session s.
2. Otherwise, search the table $\mathcal{H}$ to find values $\sigma^0_{i,j}, \sigma^1_{i,j}$ belonging to session s, for some $i, j \in [k]$. If no such pairs are found, output $\perp$ (indicating failure of the simulation). Otherwise, extract the challenge $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$ as $\sigma^0_{i,j} \oplus \sigma^1_{i,j}$, and proceed as follows.
   (a) If the next scheduled message is p1, then for each $h \in [n]$ act as follows. If $\sigma_h = 0$, act according to Step $\widehat{\text{p1}}$ of BH-protocol. Otherwise (i.e., if $\sigma_h = 1$), commit to the entries of the adjacency matrix of the complete graph K_n and to a random permutation π.
   (b) Otherwise (i.e., the next scheduled message is p2), check (in hist) that the verifier has properly decommitted to all relevant values (and that the h^{th} bit of $\sigma^0_j \oplus \sigma^1_j$ equals σ_h for all $j \in [k]$) and abort otherwise.
   For each $h \in [n]$ act as follows. If $\sigma_h = 0$, decommit to all the commitments (i.e., π and the adjacency matrix). Otherwise (i.e., if $\sigma_h = 1$), decommit only to the entries $(\pi(i), \pi(j))$ with $(i, j) \in C$ where C is an arbitrary Hamiltonian cycle in K_n.

Fig. 4. The PROVE Procedure used by SIMULATE for Stage 2 messages

Proposition 3. The probability that SIMULATE outputs $\perp$ is negligible in n.

Proof. The high-level structure of our proof follows the proof of PRS. We observe that SIMULATE outputs $\perp$ only when it tries to generate Stage 2 messages. We show in Lemma 4 that for each session, the probability of outputting $\perp$ for the first time on any thread is negligible. Since SIMULATE only runs for polynomial time, there are at most polynomial sessions and threads.[4] Therefore, we conclude using the union bound that SIMULATE outputs $\perp$ with negligible probability.

Lemma 4. *For any session s_0 and any thread l_0 (called the reference session and the reference thread), the probability that session s_0 and thread l_0 is the first time* SIMULATE *outputs $\perp$ is negligible.*

Proof. Recall that for SIMULATE to extract σ, $\mathcal{V}^*$ needs to reply to two *different* challenges (Pj) with corresponding (Vj) messages ($j \geq 1$) (after $\mathcal{V}^*$ has already

[4] We will reexamine this claim in section 5, where simulation time is (a priori) unbounded.

committed to σ). Since SIMULATE generates only polynomially many uniformly random (Pj) messages, the probability of any two challenge being identical is exponentially small in n. Therefore, it is sufficient to bound the probability conditioned on SIMULATE never repeating the same challenge.[5]

We now proceed using a *random-tape counting* argument similar to PRS. For a fixed session s_0 and thread l_0, we call a random tape ρ *bad*, if running SIMULATE with that random tape makes it output $\perp$ first on session s_0 in thread l_0. The random tape is called good otherwise. As in PRS, we show that every bad random tape can be mapped to a set of super-polynomially many good random tapes. Furthermore, this set of good random tapes is unique. Such a mapping implies that the probability of a random tape being bad is negligible. Towards this goal, we provide a mapping f that takes a bad random tape to a set of good random tapes.

To construct f, we need some properties of good and bad random tapes. We call a *slot* (i.e. a message pair $\langle(\text{Pj}), (\text{Vj})\rangle$) *good* if the verifier does not `ABORT` on this challenge. Then:

1. When SIMULATE uses a *bad* random tape, all k slots of session s_0 on thread l_0 are good. (Otherwise, SIMULATE can legitimately abort session s_0 without outputting $\perp$.)
2. A random tape is *good* if there is a good slot such that (1) it is on a non-reference thread $l \neq l_0$, (2) it occurs after $\mathcal{V}^*$ has committed to σ with message (V0) on thread l_0, and (3) it occurs before the Stage 2 message (p1) takes place on thread l_0. This good slot guarantees that SIMULATE can extract σ if needed.

Properties 1 and 2 together give the following insight: Given a bad tape, "moving" a good slot from the reference thread l_0 to a non-reference thread produces a good random tape. Moreover, the rewind-schedule of SIMULATE enables us to "swap" slots across threads by swapping segments of SIMULATE's random tape. Specifically, whenever SIMULATE splits into primary and secondary processes, the two processes share the same start state, and are simulated for the same number of steps in parallel; swapping their random tapes would swap the simulation results on the corresponding threads[6].

We define a *rewinding interval* to be a recursive execution of SIMULATE on the reference thread l_0 that contains a slot, i.e. a $\langle(\text{Pj}), (\text{Vj})\rangle$-pair, but does not contain the initial message (V0) or the Stage 2 message (p1). A *minimal rewinding interval* is defined to be a rewinding interval where none of its children intervals (i.e. smaller recursive executions of SIMULATE on l_0) contain the same slot (i.e. both (Pj) and (Vj)). Following the intuition mentioned above, swapping the randomness of a rewinding interval with its corresponding intervals on non-reference threads will generate a good tape (shown in Claim 3).

We next construct the mapping f to carry out the swapping of rewinding intervals in a structured way. Intuitively, f finds disjoint subsets of minimal

[5] As in footnote 4, we will reexamine this claim in section 5, where simulation time is unbounded.

[6] $\mathcal{V}^*$ is assumed to be deterministic.

rewinding intervals and performs the swapping operation on them. The f we use here is exactly the same mapping constructed in PRS (see Figure 5.4 of [Ros04], or the appendix for a more detailed description). Even though our simulator differs from that of PRS, the mapping f works on any simulator satisfying the following two properties: (1) Each rewinding is executed twice. (2) Any two rewindings are either disjoint or one is completely contained in the other.

We proceed to give four properties of f. Claim 1 bounds the number of random tapes produced by f based on the number of minimal rewinding intervals, while Claim 2 shows that f maps different bad tapes to disjoint sets of tapes. Both these properties of f syntactically follows by using the same proof of PRS for any simulator that satisfy the two properties mentioned above and we inherit them directly. In the following claims, ρ denotes a bad random tape.

Claim 1 (f produces many tapes). $|f(\rho)| \geq 2^{k'-d}$*, where* k' *is the number of minimal rewinding intervals and* d *is the maximum number of intervals that can overlap with each other.*

Remark: We reuse the symbol d since the maximum number of intervals that can overlap each other is just the maximum depth of recursion.

Claim 2 (f produces disjoint sets of tapes). *If* $\rho' \neq \rho$ *is another bad tape,* $f(\rho)$ *and* $f(\rho')$ *are disjoint.*

Proof. These two claims were the crux of [PRS02] [Ros04]. See Claim 5.4.12 and Claim 5.4.11 in [Ros04], for more details. We remark that Claim 1 is proved with an elaborate counting argument. Claim 2, on the other hand, is proved by constructing an "inverse" of f based on the following observation. On a bad tape, good slots appear only on the reference thread l_0. Therefore, given a tape produced by f, one can locate the minimal intervals swapped by f by searching for good slots on non-reference threads, and invert those swappings. □

In Claim 3 we show that, the tapes produced by f are good, while Claim 4 counts the number of minimal rewinding intervals. These two claims depend on how SIMULATE recursively calls itself and hence we cannot refer to PRS for the proof of these two claims; nevertheless, they do hold with respect to our simulator as we prove below.

Claim 3 (f produces good tapes). *The set* $f(\rho)\backslash\{\rho\}$ *contains only good tapes (for* SIMULATE*).*

Proof. This claim depends on the order in which simulate executes its recursive calls, since that in turn determines when σ extracted. The proof of this claim by PRS (see Claim 5.4.10 in [Ros04]) requires the main thread of the simulator to be executed after the look-ahead threads. SIMULATE, however, runs the two executions in parallel. Nevertheless, we provide an alternative proof that handles such a parallel rewinding.

Consider $\rho' \in f(\rho)$, $\rho' \neq \rho$. Let I be the first minimal rewinding interval swapped by f, and let J be the corresponding interval where I is swapped to.

Since I is the first interval to be swapped, the contents of I and J are exchanged on ρ' (while later intervals may be entirely changed due to this swap). Observe that after swapping, the $\langle(\mathrm{Pj}), (\mathrm{Vj})\rangle$ message pair that originally occurred in I will now appear on a non-reference thread inside J. Now, there are two cases depending on J:

Case 1: J does not contain the first Stage 2 message (p1) before the swap. After swapping the random tapes, (p1) would occur on the reference thread after executing both I and J. By property 2, we arrive at a good tape.

Case 2: J contains the first Stage 2 message (p1) before the swap. By the definition of a bad random tape, SIMULATE gets stuck for the first time on the reference thread after I and J are executed; Consequently, after swapping the random tape, SIMULATE will not get stuck during I. SIMULATE also cannot get stuck later on thread l_0, again due to property 2. In this case, we also arrive at a good tape. □

Claim 4. *There are at least $k' = k - 2d$ minimal rewinding intervals for session s_0 on thread l_0 (for* SIMULATE*).*

Proof. This claim depends on the number of recursive calls made by SIMULATE. For now, SIMULATE$(t, \cdot, \cdot)$ splits t into two halves just like in PRS, thus this result follows using the same proof as in PRS. Later, in Claim 7, we provide a self-contained proof of this fact in a more general setting. □

Concluding proof of Lemma 4: It follows from Claims 1, 2, 3 and 4 that every bad tape is mapped to a unique set of at least 2^{k-3d} good random tapes. Hence, the probability that a random tape is bad is at most

$$\frac{1}{2^{k-3d}}$$

Recall that $d = \lceil \log_2 T \rceil \in O(\log n)$, since T is a polynomial in n. Therefore, the probability of a bad tape occurring is negligible if $k \in \omega(\log n)$. □

This concludes the proof of Proposition 3. □

4.3 Precise Quadratic Simulation

Recall that SIMULATE takes as input t, and simulates the verifier for t steps. Since the actual simulator $\mathcal{S}_{\mathcal{V}}$ (described in Figure 5) does not know a priori the running time of the verifier, it calls SIMULATE with increasing values of $\hat{t}$, doubling every time SIMULATE returns an incomplete view. On the other hand, should SIMULATE ever output $\perp$, $\mathcal{S}_{\mathcal{V}}$ will immediately output $\perp$ as will and terminate. Also, $\mathcal{S}_{\mathcal{V}}$ runs SIMULATE with two random tapes, one of which is used exclusively whenever SIMULATE is on the main thread. Since, $\mathcal{S}_{\mathcal{V}}$ uses the same tape every time it calls SIMULATE, the view of $\mathcal{V}^*$ on the main thread proceeds identically in all the calls to SIMULATE.

$\mathcal{S}_{\mathcal{V}}$ (ρ_1, ρ_2), where ρ_1 and ρ_2 are random tapes.

1. Set $\hat{t} = 1, \mathsf{st} =$ initial state of $\mathcal{V}^*, \mathcal{H} = \emptyset$.
2. While SIMULATE did not generate a full view of $\mathcal{V}^*$:
   (a) $\hat{t} \leftarrow 2\hat{t}$
   (b) run SIMULATE$(\hat{t}, \mathsf{st}, \emptyset, (\rho_1, \rho_2))$, where random tape ρ_1 is exclusively used to simulate the verifier on the main thread, and random tape ρ_2 is used for all other threads.
   (c) output $\perp$ if SIMULATE outputs $\perp$
3. Output the full view $\mathcal{V}^*$ (i.e., random coins and messages exchanged) generated on the final run of SIMULATE$(\hat{t}, \mathsf{st}, \emptyset)$

Fig. 5. The Quadratically Precise Simulator

Lemma 5 (Concurrent Zero Knowledge). *The ensembles* $\{\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}$ $(x, z)\}_{x \in L, z \in \{0,1\}}$ *and* $\{\text{VIEW}_{\mathcal{V}}$ $(x, z)\}_{x \in L, z \in \{0,1\}}$ *are computationally indistinguishable over L.*

Proof. We consider the following "intermediate" simulator $\mathcal{S}'$ that on input x (and auxiliary input z), proceeds just like $\mathcal{S}$ (which in turn behaves like an honest prover) in order to generate messages in Stage 1 of the view. Upon entering Stage 2, $\mathcal{S}'$ outputs $\perp$ if $\mathcal{S}$ does; otherwise, $\mathcal{S}'$ proceeds as an honest prover in order to generate messages in Stage 2 of the view. Indistinguishability of the simulation by $\mathcal{S}$ then follows from the following two claims:

Claim 5. *The ensembles* $\{\text{VIEW}_{\mathcal{S}'_{\mathcal{V}}}$ $(x, z)\}_{x \in L, z \in \{0,1\}}$ *and* $\{\text{VIEW}_{\mathcal{V}}$ $(x, z)\}_{x \in L, z \in \{0,1\}}$ *are statistically close over L.*

Proof. We consider another intermediate simulator $\mathcal{S}''$ that proceeds identically like $\mathcal{S}'$ except that whenever $\mathcal{S}'$ outputs $\perp$ in a Stage 2 message, $\mathcal{S}'$ instead continues simulating like an honest prover. Essentially, $\mathcal{S}''$ never fails. Since $\mathcal{S}''$ calls SIMULATE for several values of t, this can skew the distribution. However, recall that the random tape fed by $\mathcal{S}''$ into SIMULATE to simulate the view on the main thread is identical for every call. Therefore, the view on the main thread of SIMULATE proceeds identically in every call to SIMULATE. Thus, it follows from the fact that the Stage 1 messages are generated uniform at random and that $\mathcal{S}''$ proceeds as the honest prover in Stage 2, the view output by $\mathcal{S}''$ and the view of $\mathcal{V}^*$ are identically distributed.

It remains to show that view output by $\mathcal{S}'$ and $\mathcal{S}''$ are statistically close over L. The only difference between $\mathcal{S}'$ and $\mathcal{S}''$ is that $\mathcal{S}'$ outputs $\perp$ sometimes. It suffices to show that $\mathcal{S}'$ outputs $\perp$ with negligible probability. From Proposition 3, we know that SIMULATE outputs $\perp$ only with negligible probability. Since SIMULATE is called at most logarithmically many times due to the doubling of t, using the union bound we conclude that $\mathcal{S}'$ outputs $\perp$ with negligible probability. $\square$

Claim 6. *The ensembles* $\{\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}(x,z)\}_{x\in L,z\in\{0,1\}}$ *and* $\{\text{VIEW}_{\mathcal{S}'_{\mathcal{V}}}(x,z)\}_{x\in L,z\in\{0,1\}}$ *are computationally indistinguishable over* L.

Proof. The only difference between $\mathcal{S}$ and $\mathcal{S}'$ is in the manner in which the Stage 2 messages are generated. Indistinguishability follows from the special honest-verifier ZK property using a standard hybrid argument, as given below.

Assume for contradiction that there exists a verifier V^*, a distinguisher D and a polynomial $p(\cdot)$ such that D distinguishes the ensembles $\{\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}(x,z)\}$ and $\{\text{VIEW}_{\mathcal{S}'_{\mathcal{V}}}(x,z)\}$ with probability $\frac{1}{p(n)}$. Furthermore, let the running time of V^* be bounded by some polynomial $q(n)$. We consider a sequence of hybrid simulators, S_i for $i = 0$ to $q(n)$. S_i proceeds exactly like S, with the exception that in the first i proofs that reach the second stage, it proceeds using the honest prover strategy in the second stage for those proofs. By construction $S_0 = S$ and $S_{q(n)} = \mathcal{S}'$ (since there are at most $q(n)$ sessions, bounded by the running time of the simulators). By assumption the output of S_0 and $S_{q(n)}$ are distinguishable with probability $\frac{1}{p(n)}$, so there must exist some j such that the output of S_j and S_{j+1} are distinguishable with probability $\frac{1}{p(n)q(n)}$. Furthermore, since S_j proceeds exactly as S_{j+1} in the first j sessions that reach the second stage, and by construction they proceed identically in the first stage in all sessions, there exists a partial view v of S_j and S_{j+1}—which defines an instance for the protocol in the second stage of the $j+1$ session—such that the output of S_j and S_{j+1} are distinguishable, conditioned on the event that S_j and S_{j+1} feed V^* the view v. Since the only difference between the view of V^* in S_j and S_{j+1} is that the former is a simulated view, while the later is a view generated using an honest prover strategy, this contradicts the special honest-verifier ZK property of the BH-protocol in the second stage of the protocol. □

Lemma 6 (Quadratic Precision). *Let* $\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}$ *be the output of the simulator* $\mathcal{S}_{\mathcal{V}}$ *, and* t *be the running time of* $\mathcal{V}^*$ *on the view* $\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}$ *. Then,* $\mathcal{S}_{\mathcal{V}}$ *runs in time* $O(t^2)$.

Proof. Recall that, $\mathcal{S}_{\mathcal{V}}$ runs SIMULATE with increasing values of $\hat{t}$, doubling each time, until a view is output. We again use the fact that the view on the main thread of SIMULATE proceeds identically (in this case, proceeds as $\text{VIEW}_{\mathcal{S}_{\mathcal{V}}}$) since the random tape used to simulate the main thread is identical in every call to SIMULATE. Therefore, the final value of $\hat{t}$ when v is output satisfies,

$$t \leq \hat{t} < 2t$$

The running time of $\mathcal{S}_{\mathcal{V}}$ is simply the sum of the running times of $\text{SIMULATE}(t, \mathsf{st}, \emptyset)$ with $t = 1, 2, 4, \ldots, \hat{t}$. By Lemma 2, this running time is bounded by

$$c1^2 + c2^2 + c4^2 + \cdots + c\hat{t}^2 \leq 2c\hat{t}^2 \leq 8ct^2$$

For some constant c. □

4.4 Improved Precision

We now consider a generalized version of SIMULATE. Let $g \geq 2$ be an integer; $\text{SIMULATE}_g(t, \cdot, \cdot)$ will now divide t in g smaller intervals. If t does not divide into g evenly, that is if $t = qg + r$ with $r > 0$, let the first r sub-intervals have length $\lfloor t/g \rfloor + 1$, and the rest of the $g - r$ sub-intervals have length $\lfloor t/g \rfloor$. We call g the splitting factor, and assume $k/g \in \omega(\log n)$ as stated in Theorem 1. Due to the lack of space most of the details of this section are given in the full version [PPS+07]. We only state our main claims here.

In the full version, we demonstrate that the running time of our new simulator is given by the following lemma.

Lemma 7 (Improved Precision). *Let* $\text{VIEW}_{\mathcal{S}_\mathcal{V}}$ *be the output of the simulator* $\mathcal{S}_\mathcal{V}$ *using* SIMULATE_g*, and* t *be the running time of* $\mathcal{V}^*$ *on the view* $\text{VIEW}_{\mathcal{S}_\mathcal{V}}$*. Then,* $\mathcal{S}_\mathcal{V}$ *runs in time* $O(t \cdot 2^{\log_g t}) = O(t^{1+\log_g 2})$*.*

Thereafter, we show there the indistinguishability of the simulator's output.

Lemma 8 (Concurrent Zero Knowledge). $\{\text{VIEW}_{\mathcal{S}_\mathcal{V}}(x, z)\}_{x \in L, z \in \{0,1\}}$ *and* $\{\text{VIEW}_\mathcal{V}(x, z)\}_{x \in L, z \in \{0,1\}}$ *are computationally indistinguishable over* L*.*

Finally, in order to deduce our main lemma, we demonstrate the following important claim regarding the number of minimum intervals with respect to our new simulator. This claim is analogous to claim 4. It, however, depends on the splitting factor g and is modified as follows:

Claim 7 (Number of Minimal Rewinding Intervals). *There are at least* $k' = \frac{k}{g-1} - 2d$ *minimal rewinding intervals for session* s_0 *on thread* l_0 *(for* SIMULATE_g*), where* d *is the recursion depth.*

The main use of this lemma is in deducing that our new simulator outputs $\perp$ with only negligible probability.

4.5 Proof of Main Lemma and Consequences

Lemma 9 (Main Lemma). *Assuming the existence of one-way functions, then for every* $k, g \in \mathbb{N}$ *such that* $k/g \in \omega(\log n)$*, there exists an* $O(k)$*-round concurrent zero knowledge argument with precision* $p(t) \in O(t \cdot 2^{\log_g t})$ *for all languages in* $\mathcal{NP}$*.*

Proof. Using Lemmata 7 and 8, we conclude that the simulator $\mathcal{S}_\mathcal{V}$ (using SIMULATE_g) outputs a verifier view of the right distribution with precision $O(t \cdot 2^{\log_g t})$. □

By setting $g = 2^{1/\varepsilon}$ and $k \in \omega(\log n)$ in our main lemma, we get our first theorem.

Theorem 4. *Assuming the existence of one-way functions, for every* $\varepsilon > 0$*, there exists a* $\omega(\log n)$*-round concurrent zero knowledge argument for all languages in* $\mathcal{NP}$ *with precision* $p(t) = O(t^{1+\varepsilon})$*.*

Finally, by setting $g = n^{\epsilon/2}$ and $k = n^{\epsilon}$ in our main lemma, we get our next theorem.

Theorem 5. *Assuming the existence of one-way functions, for every* $\varepsilon > 0$, *there exists an* $O(n^{\varepsilon})$*-round concurrent zero knowledge argument for all languages in* $\mathcal{NP}$ *with precision* $p(t) = O(t2^{\frac{2}{\epsilon}\log_n t})$. *As a corollary, we obtain the following: For every* $\epsilon > 0$, *there exists an* $O(n^{\varepsilon})$*-round protocol* $\langle P, V\rangle$ *such that for every* $c > 0$, $\langle P, V\rangle$ *is a concurrent zero knowledge argument with precision* $p(n,t) = O(t)$ *with respect to verifiers with running time bounded by* n^c *for all languages in* $\mathcal{NP}$.

5 Statistically Precise Concurrent Zero-Knowledge

In this section, we construct a statistically precise concurrent ZK argument for all of $\mathcal{NP}$. Recall that statistically precise ZK requires the simulation of all malicious verifiers (even those having a priori unbounded running time) and the distribution of the simulated view must be statistically close to that of the real view. A first approach is to use the previous protocol and simulator with the splitting factor fixed appropriately. However this approach does not work directly; briefly the reason being that we will need k to superpolynomial in n. We thus present a slightly modified simulator, which appears shortly.

Theorem 6. *Assume the existence of* claw-free permutations, *then there exists a poly*(n)*-round statistically precise concurrent zero-knowledge argument for all of* $\mathcal{NP}$ *with precision* $p(n,t) = O(t^{1+\frac{1}{\log n}})$.

Description of protocol: We essentially use the same protocol described in Section 3 setting the number of rounds $k = 5n^2 \log n$ (n is the security parameter), with the following exception: In Stage 2 of the protocol, the prover uses perfectly hiding commitments in the BH-protocol instead of computational hiding. This makes the BH-protocol perfect-special-HVZK.

Description of simulator S: The simulator S executes $\mathcal{S}_{\mathcal{V}}$ with $g = n$ and outputs whatever $\mathcal{S}_{\mathcal{V}}$ outputs, with the following exception: while executing SIMULATE$_n$ (inside $\mathcal{S}_{\mathcal{V}}$), if the verifier in the main thread runs for more than $2^{n \log_2 n}$ steps, it terminates the execution of SIMULATE$_n$ and retrieves the partial history hist simulated in the main thread so far. Then, it continues to simulate the verifier from hist in a "straight-line" fashion—it generates uniformly random messages for the Stage 1 of the protocol, and when it reaches Stage 2 of the protocol for some session, it runs the brute-force-PROVE procedure, given in the full version. An analysis of this simulator appears in [PPS+07].

Acknowledgements. We would like to thank Alon Rosen for several helpful discussions.

References

Blu87. Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians, pp. 1444–1451 (1987)

DNS98. Dwork, C., Naor, M., Sahai, A.: Concurrent zero knowledge. In: Proc. 30th STOC, pp. 409–418 (1998)

GMR85. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proc. 17th STOC, pp. 291–304 (1985)

GMW87. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: ACM (ed.) Proc. 19th STOC, pp. 218–229 (1987); See [Gol04, Chap. 7] for more details

Gol01. Goldreich, O.: Foundations of Cryptography, vol. Basic Tools. Cambridge University Press, Cambridge (2001)

HILL99. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999); Preliminary versions appeared in STOC 1989 and STOC 1990

KP01. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in polylogarithm rounds. In: Proc. 33th STOC, pp. 560–569 (2001); Preliminary full version published as cryptology ePrint report 2000/013

MP06. Micali, S., Pass, R.: Local zero knowledge. In: Kleinberg, J.M. (ed.) STOC, pp. 306–315. ACM Press, New York (2006)

Nao91. Naor, M.: Bit commitment using pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991); Preliminary version in CRYPTO 1989

Pas06. Pass, R.: A Precise Computational Approach to Knowledge. PhD thesis, MIT (July 2006)

PPS+07. Pandey, O., Pass, R., Sahai, A., Tseng, W.-L.D., Venkitasubramaniam, M.: Precise concurrent zero knowledge. Cryptology ePrint Archive, Report 2007/451 (2007), http://eprint.iacr.org/2007/451.pdf

PRS02. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: Proc. 43rd FOCS (2002)

RK99. Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–432. Springer, Heidelberg (1999)

Ros04. Rosen, A.: The Round-Complexity of Black-Box Concurrent Zero-Knowledge. PhD thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel (2004)

Efficient Non-interactive Proof Systems for Bilinear Groups*

Jens Groth[1,**] and Amit Sahai[2,***]

[1] University College London
j.groth@ucl.ac.uk
[2] University of California Los Angeles
sahai@cs.ucla.edu

Abstract. Non-interactive zero-knowledge proofs and non-interactive witness-indistinguishable proofs have played a significant role in the theory of cryptography. However, lack of efficiency has prevented them from being used in practice. One of the roots of this inefficiency is that non-interactive zero-knowledge proofs have been constructed for general NP-complete languages such as Circuit Satisfiability, causing an expensive blowup in the size of the statement when reducing it to a circuit. The contribution of this paper is a general methodology for constructing very simple and efficient non-interactive zero-knowledge proofs and non-interactive witness-indistinguishable proofs that work directly for groups with a bilinear map, without needing a reduction to Circuit Satisfiability.

Groups with bilinear maps have enjoyed tremendous success in the field of cryptography in recent years and have been used to construct a plethora of protocols. This paper provides non-interactive witness-indistinguishable proofs and non-interactive zero-knowledge proofs that can be used in connection with these protocols. Our goal is to spread the use of non-interactive cryptographic proofs from mainly theoretical purposes to the large class of practical cryptographic protocols based on bilinear groups.

Keywords: Non-interactive witness-indistinguishability, non-interactive zero-knowledge, common reference string, bilinear groups.

1 Introduction

Non-interactive zero-knowledge proofs and non-interactive witness-indistinguishable proofs have played a significant role in the theory of cryptography. However, lack of efficiency has prevented them from being used in practice. Our goal is to construct efficient and practical non-interactive zero-knowledge (NIZK) proofs and non-interactive witness-indistinguishable (NIWI) proofs.

* Work presented and part of work done while participating in Securing Cyberspace: Applications and Foundations of Cryptography and Computer Security, Institute of Pure and Applied Mathematics, UCLA, 2006.

** Part of work done while at UCLA supported by NSF ITR/Cybertrust grant 0456717.

*** This research was supported in part by NSF ITR and Cybertrust programs (including grants 0627781, 0456717, 0716389, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Okawa Research Award, and an Alfred P. Sloan Foundation Research Fellowship.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 415–432, 2008.

Blum, Feldman and Micali [4] introduced NIZK proofs. Their paper and subsequent work, e.g. [19,16,29,17], demonstrates that NIZK proofs exist for all of NP. Unfortunately, these NIZK proofs are all very inefficient. While leading to interesting theoretical results, such as the construction of public-key encryption secure against chosen-ciphertext attack by Dolev, Dwork and Naor [18], they have therefore not had any impact in practice.

Since we want to construct NIZK proofs that can be used in practice, it is worthwhile to identify the roots of the inefficiency in the above mentioned NIZK proofs. One drawback is that they were designed with a general NP-complete language in mind, e.g. Circuit Satisfiability. In practice, we want to prove statements such as "the ciphertext c encrypts a signature on the message m" or "the three commitments c_a, c_b, c_c contain messages a, b, c so $c = ab$". An NP-reduction of even very simple statements like these gives us big circuits containing thousands of gates and the corresponding NIZK proofs become very large.

While we want to avoid an expensive NP-reduction, it is still desirable to have a general way to express statements that arise in practice instead of having to construct non-interactive proofs on an ad hoc basis. A useful observation in this context is that many public-key cryptography protocols are based on finite abelian groups. If we can capture statements that express relations between group elements, then we can express statements that come up in practice such as "the commitments c_a, c_b, c_c contain messages so $c = ab$" or "the plaintext of c is a signature on m", as long as those commitment, encryption, and signature schemes work over the same finite group. In the paper, we will therefore construct NIWI and NIZK proofs for *group-dependent* languages.

The next issue to address is where to find suitable group-dependent languages. We will look at statements related to groups with a bilinear map, which have become widely used in the design of cryptographic protocols. Not only have bilinear groups been used to give new constructions of such cryptographic staples as public-key encryption, digital signatures, and key agreement (see [31] and the references therein), but bilinear groups have enabled the first constructions achieving goals that had never been attained before. The most notable of these is the Identity-Based Encryption scheme of Boneh and Franklin [10] (see also [6,7,35]), and there are many others, such as Attribute-Based Encryption [32,22], Searchable Public-Key Encryption [9,12,13], and One-time Double-Homomorphic Encryption [11]. For an incomplete list of papers (currently over 200) on the application of bilinear groups in cryptography, see [2].

1.1 Our Contribution

For completeness, let us recap the definition of a bilinear group. *Please note that for notational convenience we will follow the tradition of mathematics and use additive notation*[1] *for the binary operations in G_1 and G_2.* We have a probabilistic

[1] We remark that in the cryptographic literature it is more common to use multiplicative notation for these groups, since the "discrete log problem" is believed to be hard in these groups, which is also important to us. In our setting, however, it will be much more convenient to use multiplicative notation to refer to the action of the bilinear map.

polynomial time algorithm $\mathcal{G}$ that takes a security parameter as input and outputs $(\mathbf{n}, G_1, G_2, G_T, e, \mathcal{P}_1, \mathcal{P}_2)$ where

- G_1, G_2, G_T are descriptions of cyclic groups of order $\mathbf{n}$.
- The elements $\mathcal{P}_1, \mathcal{P}_2$ generate G_1 and G_2 respectively.
- $e : G_1 \times G_2$ is a non-degenerate bilinear map so $e(\mathcal{P}_1, \mathcal{P}_2)$ generates G_T and for all $a, b \in \mathbb{Z}_\mathbf{n}$ we have $e(a\mathcal{P}_1, b\mathcal{P}_2) = e(\mathcal{P}_1, \mathcal{P}_2)^{ab}$.
- We can efficiently compute group operations, compute the bilinear map and decide membership.

In this work, we develop a general set of highly efficient techniques for proving statements involving bilinear groups. The generality of our work extends in two directions. First, we formulate our constructions in terms of modules over commutative rings with an associated bilinear map. This framework captures all known bilinear groups with cryptographic significance – for both supersingular and ordinary elliptic curves, for groups of both prime and composite order. Second, we consider all mathematical operations that can take place in the context of a bilinear group - addition in G_1 and G_2, scalar point-multiplication, addition or multiplication of scalars, and use of the bilinear map. We also allow both group elements and exponents to be "unknowns" in the statements to be proven.

With our level of generality, it would for example be easy to write down a short statement, using the operations above, that encodes "c is an encryption of the value committed to in d under the product of the two keys committed to in a and b" where the encryptions and commitments being referred to are existing cryptographic constructions based on bilinear groups. Logical operations like AND and OR are also easy to encode into our framework using standard techniques in arithmetization.

The proof systems we build are *non-interactive*. This allows them to be used in contexts where interaction is undesirable or impossible. We first build highly efficient witness-indistinguishable proof systems, which are of independent interest. We then show how to transform these into zero-knowledge proof systems. We also provide a detailed examination of the efficiency of our constructions in various settings (depending on what type of bilinear group is used).

The security of constructions arising from our framework can be based on *any* of a variety of computational assumptions about bilinear groups (3 of which we discuss in detail here). Thus, our techniques do not rely on any one assumption in particular.

Informal Statement of Our Results. We consider equations over variables from G_1, G_2 and $\mathbb{Z}_\mathbf{n}$ as described in Figure 1. We construct efficient witness-indistinguishable proofs for the simultaneous satisfiability of a set of such equations. The witness-indistinguishable proofs have perfect completeness and there are two computationally indistinguishable types of common reference strings giving respectively perfect soundness and perfect witness indistinguishability. Due to lack of space we have to refer to the full paper [28] for precise definitions.

We also consider the question of non-interactive zero-knowledge. We show that we can give zero-knowledge proofs for multi-scalar multiplication in G_1 or G_2 and for quadratic equations in $\mathbb{Z}_\mathbf{n}$. We can also give zero-knowledge proofs for pairing product equations with $t_T = 1$. When $t_T \neq 1$ we can still give zero-knowledge proofs if we can find $\mathcal{P}_1, \mathcal{Q}_1, \dots, \mathcal{P}_n, \mathcal{Q}_n$ such that $t_T = \prod_{i=1}^n e(\mathcal{P}_i, Q_i)$.

Variables: $\mathcal{X}_1,\ldots,\mathcal{X}_m \in G_1$, $\mathcal{Y}_1,\ldots,\mathcal{Y}_n \in G_2$, $x_1,\ldots,x_{m'},y_1,\ldots,y_{n'} \in \mathbb{Z}_{\mathbf{n}}$. [a]
Pairing product equation:

$$\prod_{i=1}^{n} e(\mathcal{A}_i,\mathcal{Y}_i)\cdot\prod_{i=1}^{m} e(\mathcal{X}_i,\mathcal{B}_i)\cdot\prod_{i=1}^{m}\prod_{j=1}^{n} e(\mathcal{X}_i,\mathcal{Y}_j)^{\gamma_{ij}} = t_T ,$$

for constants $\mathcal{A}_i \in G_1, \mathcal{B}_i \in G_2, t_T \in G_T, \gamma_{ij} \in \mathbb{Z}_{\mathbf{n}}$.
Multi-scalar multiplication equation in G_1:

$$\sum_{i=1}^{n'} y_i\mathcal{A}_i + \sum_{i=1}^{m} b_i\mathcal{X}_i + \sum_{i=1}^{m}\sum_{j=1}^{n'} \gamma_{ij}y_j\mathcal{X}_i = \mathcal{T}_1,$$

for constants $\mathcal{A}_i, \mathcal{T}_1 \in G_1$ and $b_i, \gamma_{ij} \in \mathbb{Z}_{\mathbf{n}}$. [b]
Multi-scalar multiplication equation in G_2:

$$\sum_{i=1}^{n} a_i\mathcal{Y}_i + \sum_{i=1}^{m'} x_i\mathcal{B}_i + \sum_{i=1}^{m'}\sum_{j=1}^{n} \gamma_{ij}x_i\mathcal{Y}_j = \mathcal{T}_2,$$

for constants $\mathcal{B}_i, \mathcal{T}_2 \in G_2$ and $a_i, \gamma_{ij} \in \mathbb{Z}_{\mathbf{n}}$.
Quadratic equation in $\mathbb{Z}_{\mathbf{n}}$:

$$\sum_{i=1}^{n'} a_iy_i + \sum_{i=1}^{m'} x_ib_i + \sum_{i=1}^{m'}\sum_{j=1}^{n'} \gamma_{ij}x_iy_j = t,$$

for constants $a_i, \gamma_{ij}, t \in \mathbb{Z}_{\mathbf{n}}$.

[a] We list variables in $\mathbb{Z}_{\mathbf{n}}$ in two separate groups because we will treat them differently in the NIWI proofs. If we wish to deal with only one group of variables in $\mathbb{Z}_{\mathbf{n}}$ we can add equations in $\mathbb{Z}_{\mathbf{n}}$ of the form $x_1 = y_1, x_2 = y_2$, etc.
[b] With multiplicative notation, these equations would be multi-exponentiation equations. We use additive notation for G_1 and G_2, since this will be notationally convenient in the paper, but stress that the discrete logarithm problem will typically be hard in these groups.

Fig. 1. Equations over groups with bilinear map

Instantiations. In the full paper we give three possible instantiations of the bilinear groups; there are many more. The first instantiation is based on the composite order groups introduced by Boneh, Goh and Nissim [11]. We work over a composite order bilinear group $(\mathbf{n}, G, G_T, e, \mathcal{P})$ where $\mathbf{n} = \mathbf{pq}$. The security of this instantiation is based on the subgroup decision assumption that says it is hard to distinguish random elements of order $\mathbf{n}$ from random elements of order $\mathbf{q}$.

The second instantiation is based on prime order groups $(\mathbf{p}, G_1, G_2, G_T, e, \mathcal{P}_1, \mathcal{P}_2)$. Security depends on the symmetric external Diffie-Hellman (SXDH) assumption [33,8,1,20,34] that says the DDH problem is hard in both G_1 and G_2.

The third instantiation is based on prime order groups $(\mathbf{p}, G, G_T, e, \mathcal{P})$ where the decisional linear (DLIN) problem is hard. The DLIN problem introduced by

Boneh, Boyen and Shacham [8] states that given $(\alpha\mathcal{P}, \beta\mathcal{P}, r\alpha\mathcal{P}, s\beta\mathcal{P}, t\mathcal{P})$ for random $\alpha, \beta, r, s \in \mathbb{Z}_{\mathbf{p}}$ it is hard to tell whether $t = r + s$ or t is random.

The instantiations illustrate the variety of ways bilinear groups can be constructed. We can choose prime order or composite order groups, we can use $G_1 = G_2$ and $G_1 \neq G_2$, and we can make various cryptographic assumptions. All three security assumptions have been used in the cryptographic literature to build interesting protocols.

For all three instantiations, the techniques presented here give us short NIWI proofs. In particular, the cost in proof size of each extra equation is constant and independent of the number of variables in the equation. The size of the proofs, can be computed by adding the cost, measured in group elements from G_1 or G_2, of each variable and each equation listed in Figure 2. We refer to the full paper [28] for more detailed tables.

	Subgroup decision	SXDH	DLIN
Variable in G_1 or G_2	1	2	3
Variable in $\mathbb{Z}_{\mathbf{n}}$ or $\mathbb{Z}_{\mathbf{p}}$	1	2	3
Paring product equation	1	8	9
Multi-scalar multiplication in G_1 or G_2	1	6	9
Quadratic equation in $\mathbb{Z}_{\mathbf{n}}$ or $\mathbb{Z}_{\mathbf{p}}$	1	4	6

Fig. 2. Number of group elements each variable or equation adds to the size of a NIWI proof

1.2 Related Work

As we mentioned before, early work on NIZK proofs demonstrated that all NP-languages have non-interactive proofs, however, did not yield efficient proofs. One cause for these proofs being inefficient in practice was the need for an expensive NP-reduction to e.g. Circuit Satisfiability. Another cause of inefficiency was the reliance on the so-called hidden bits model, which even for small circuits is inefficient.

Groth, Ostrovsky, and Sahai [27,26] investigated NIZK proofs for Circuit Satisfiability using bilinear groups. This addressed the second cause of inefficiency since their techniques give efficient proofs for Circuit Satisfiability, but to use their proofs one must still make an NP-reduction to Circuit Satisfiability thus limiting the applications. We stress that while [27,26] used bilinear groups, their application was to build proof systems for Circuit Satisfiability. Here, we devise entirely new techniques to deal with general statements about equations in bilinear groups, *without* having to reduce to an NP-complete language.

Addressing the issue of avoiding an expensive NP-reduction we have works by Boyen and Waters [13,14] that suggest efficient NIWI proofs for statements related to group signatures. These proofs are based on bilinear groups of composite order and rely on the subgroup decision assumption.

Groth [23] was the first to suggest a general group-dependent language and NIZK proofs for statements in this language. He investigated satisfiability of pairing product equations and only allowed group elements to be variables. He also looked only at the special case of prime order groups G, G_T with a bilinear map $e : G \times G \to G_T$ and, based on the decisional linear assumption [8], constructed NIZK proofs for such pairing product equations. However, even for very small statements, the very different and much more complicated techniques of Groth yield proofs consisting of thousands of group elements (whereas ours would be in the tens). Our techniques are much easier to understand, significantly more general, and vastly more efficient.

We summarize our comparison with other works on NIZK proofs in Figure 3.

	Inefficient	Efficient
Circuit Satisfiability	E.g. [29]	[27,26]
Group-dependent language	[23] (restricted case)	This work

Fig. 3. Classification of NIZK proofs according to usefulness

We note that there have been many earlier works (starting with [21]) dealing with efficient *interactive* zero-knowledge protocols for a number of algebraic relations. Here, we focus on *non-interactive* proofs. We also note that even for interactive zero-knowledge proofs, no set of techniques was known for dealing with general algebraic assertions arising in bilinear groups, as we do here.

1.3 New Techniques

[27,26,23] start by constructing non-interactive proofs for simple statements and then combine many of them to get more powerful proofs. The main building block in [27], for instance, is a proof that a given commitment contains either 0 or 1, which has little expressive power on its own. Our approach is the opposite: we directly construct proofs for very expressive languages; as such, our techniques are very different from previous work.

The way we achieve our generality is by viewing the groups G_1, G_2, G_T as modules over the ring $\mathbb{Z}_\mathbf{n}$. The ring $\mathbb{Z}_\mathbf{n}$ itself can also be viewed as a $\mathbb{Z}_\mathbf{n}$-module. We therefore look at the more general question of satisfiability of quadratic equations over $\mathbb{Z}_\mathbf{n}$-modules A_1, A_2, A_T with a bilinear map, see Section 2 for details. Since many bilinear groups with various cryptographic assumptions and various mathematical properties can be viewed as modules we are not bound to any particular bilinear group or any particular assumption. We remark that while bilinear groups can be interpreted as modules with a bilinear map, it is possible that there exist other interesting modules with a bilinear map that are not based on bilinear groups. We leave the existence of such modules as an interesting open problem.

Given modules A_1, A_2, A_T with a bilinear map, we construct new modules B_1, B_2, B_T, also equipped with a bilinear map, and we map the elements in A_1, A_2, A_T into B_1, B_2, B_T. More precisely, we devise commitment schemes that map variables from A_1, A_2 to the modules B_1, B_2. The commitment schemes are homomorphic with respect to the module operations but also with respect to the bilinear map.

Our techniques for constructing witness-indistinguishable proofs are fairly involved mathematically, but we will try to present some high level intuition here. (We give more detailed intuition later in Section 5, where we present our main proof system). The main idea is the following: because our commitment schemes are homomorphic *and* we equip them with a bilinear map, we can take the equation that we are trying to prove, and just replace the variables in the equation with commitments to those variables. Of course, because the commitment schemes are hiding, the equations will no longer be valid. Intuitively, however, we can extract out the additional terms introduced by the randomness of the commitments: if we give away these terms in the proof, then this would be a *convincing* proof of the equation's validity (again, because of the homomorphic properties). But, giving away these terms might destroy witness indistinguishability. Suppose, however, that there is only one "additional term" introduced by substituting the commitments. Then, because it would be the unique value which makes the equation true, giving it away would preserve witness indistinguishability! In general, we are not so lucky. But if there are many terms, that means that these terms are not unique, and because of the nice algebraic environment that we work in, we can randomize these terms so that the equation is still true, but so that we effectively reduce to the case of there being a single term being given away with a unique value.

1.4 Applications

Independently of our work, Boyen and Waters [14] have constructed non-interactive proofs that they use for group signatures (see also their earlier paper [13]). These proofs can be seen as examples of the NIWI proofs in instantiation 1. Subsequent to the announcement of our work, several papers have built upon it: Chandran, Groth and Sahai [15] have constructed ring-signatures of sub-linear size using the NIWI proofs in the first instantiation, which is based on the subgroup decision problem. Groth and Lu [25] have used the NIWI and NIZK proofs from instantiation 3 to construct a NIZK proof for the correctness of a shuffle. Groth [24] has used the NIWI and NIZK proofs from instantiation 3 to construct a fully anonymous group signature scheme. Belenkiy, Chase, Kohlweiss and Lysyanskaya [3] have used instantiations 2 and 3 to construct non-interactive anonymous credentials. Also, by attaching NIZK proofs to semantically secure public-key encryption in any instantiation we get an efficient non-interactive verifiable cryptosystem. Boneh [5] has suggested using this for optimistic fair exchange [30], where two parties use a trusted but lazy third party to guarantee fairness.

2 Modules with Bilinear Maps

Let $(\mathcal{R}, +, \cdot, 0, 1)$ be a finite commutative ring. Recall that an $\mathcal{R}$-module A is an abelian group $(A, +, 0)$ where the ring acts on the group such that $\forall r, s \in \mathcal{R}\ \forall x, y \in A$:

$$(r+s)x = rx + sx \ \wedge\ r(x+y) = rx + ry \ \wedge\ r(sx) = (rs)x \ \wedge\ 1x = x.$$

A cyclic group G of order $\mathbf{n}$ can in a natural way be viewed as a $\mathbb{Z}_{\mathbf{n}}$-module. We will observe that all the equations in Figure 1 can be viewed as equations over $\mathbb{Z}_{\mathbf{n}}$-modules with a bilinear map. To generalize completely, let $\mathcal{R}$ be a finite commutative ring and let A_1, A_2, A_T be finite $\mathcal{R}$-modules with a bilinear map $f : A_1 \times A_2 \to A_T$. We consider quadratic equations over variables $x_1, \ldots, x_m \in A_1, y_1, \ldots, y_n \in A_2$ of the form

$$\sum_{j=1}^{n} f(a_j, y_j) + \sum_{i=1}^{m} f(x_i, b_i) + \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_{ij} f(x_i, y_j) = t.$$

In order to simplify notation, let us for $x_1, \ldots, x_n \in A_1, y_1, \ldots, y_n \in A_2$ define

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} f(x_i, y_i).$$

The equations can now be written as

$$\boldsymbol{a} \cdot \boldsymbol{y} + \boldsymbol{x} \cdot \boldsymbol{b} + \boldsymbol{x} \cdot \Gamma \boldsymbol{y} = t.$$

We note for future use that due to the bilinear properties of f, we have for any matrix $\Gamma \in \mathrm{Mat}_{m \times n}(\mathcal{R})$ and for any $x_1, \ldots, x_m, y_1, \ldots, y_n$ that $\boldsymbol{x} \cdot \Gamma \boldsymbol{y} = \Gamma^\top \boldsymbol{x} \cdot \boldsymbol{y}$.

Let us now return to the equations in Figure 1 and see how they can be recast as quadratic equations over $\mathbb{Z}_{\mathbf{n}}$-modules with a bilinear map.

Pairing product equations: Define $\mathcal{R} = \mathbb{Z}_{\mathbf{n}}, A_1 = G_1, A_2 = G_2, A_T = G_T, f(x, y) = e(x, y)$ and we can rewrite[2] the pairing product equation as $(\boldsymbol{\mathcal{A}} \cdot \boldsymbol{\mathcal{Y}})(\boldsymbol{\mathcal{X}} \cdot \boldsymbol{\mathcal{B}})(\boldsymbol{\mathcal{X}} \cdot \Gamma \boldsymbol{\mathcal{Y}}) = t_T$.

Multi-scalar multiplication in G_1: Define $\mathcal{R} = \mathbb{Z}_{\mathbf{n}}, A_1 = G_1, A_2 = \mathbb{Z}_{\mathbf{n}}, A_T = G_1, f(\mathcal{X}, y) = y\mathcal{X}$ and we can rewrite the multi-scalar multiplication equation as $\boldsymbol{\mathcal{A}} \cdot \boldsymbol{y} + \boldsymbol{\mathcal{X}} \cdot \boldsymbol{b} + \boldsymbol{\mathcal{X}} \cdot \Gamma \boldsymbol{y} = \mathcal{T}_1$.

Multi-scalar multiplication in G_2: Define $\mathcal{R} = \mathbb{Z}_{\mathbf{n}}, A_1 = \mathbb{Z}_{\mathbf{n}}, A_2 = G_2, A_T = G_2, f(x, \mathcal{Y}) = x\mathcal{Y}$ and we can rewrite the multi-scalar multiplication equation as $\boldsymbol{a} \cdot \boldsymbol{\mathcal{Y}} + \boldsymbol{x} \cdot \boldsymbol{\mathcal{B}} + \boldsymbol{x} \cdot \Gamma \boldsymbol{\mathcal{Y}} = \mathcal{T}_2$.

Quadratic equation in $\mathbb{Z}_{\mathbf{n}}$: Define $\mathcal{R} = \mathbb{Z}_{\mathbf{n}}, A_1 = \mathbb{Z}_{\mathbf{n}}, A_2 = \mathbb{Z}_{\mathbf{n}}, A_T = \mathbb{Z}_{\mathbf{n}}, f(x, y) = xy \bmod \mathbf{n}$ and we can rewrite the quadratic equation in $\mathbb{Z}_{\mathbf{n}}$ as $\boldsymbol{a} \cdot \boldsymbol{y} + \boldsymbol{x} \cdot \boldsymbol{b} + \boldsymbol{x} \cdot \Gamma \boldsymbol{y} = t$.

From now on, we will therefore focus on the more general problem of constructing non-interactive composable witness-indistinguishable proofs for satisfiability of quadratic equations over $\mathcal{R}$-modules A_1, A_2, A_T (using additive notation for all modules) with a bilinear map f.

3 Commitment from Modules

In our NIWI proofs we will commit to the variables $x_1, \ldots, x_m \in A_1, y_1, \ldots, y_n \in A_2$. We do this by mapping them into other $\mathcal{R}$-modules B_1, B_2 and making the commitments in those modules.

Let us for now just consider how to commit to elements from one $\mathcal{R}$-module A. The public key for the commitment scheme will describe another $\mathcal{R}$-module B and $\mathcal{R}$-linear maps $\iota : A \to B$ and $p : B \to A$. It will also contain elements $u_1, \ldots, u_n \in B$. To commit to $x \in A$ we pick $r_1, \ldots, r_n \leftarrow \mathcal{R}$ at random and compute the commitment

$$c := \iota(x) + \sum_{i=1}^{n} r_i u_i.$$

Our commitment scheme will have two types of commitment keys.

[2] We use multiplicative notation here, because, usually G_T is written multiplicatively in the literature. When we work with the abstract modules, however, we will use additive notation.

Hiding key: A hiding key contains $(B, \iota, p, u_1, \ldots, u_n)$ such that $\iota(G) \subseteq \langle u_1, \ldots, u_n \rangle$. The commitment $c := \iota(x) + \sum_{i=1}^{n} r_i u_i$ is perfectly hiding when $r_1, \ldots, r_n$ are chosen at random from $\mathcal{R}$.

Binding key: A binding key contains $(B, \iota, p, u_1, \ldots, u_n)$ such that $\forall i : p(u_i) = 0$ and $\iota \circ p$ is the identity.[3] The commitment $c := \iota(x) + \sum_{i=1}^{n} r_i u_i$ is perfectly binding, since it determines x as $p(c) = p(\iota(x)) = x$.[4]

Computational indistinguishability: The main assumption that we will be making throughout this paper is that the distribution of hiding keys and the distribution of binding keys are computationally indistinguishable. Witness-indistinguishability of our NIWI proofs and later the zero-knowledge property of our NIZK proofs will rely on this property.

Often we will commit to many elements at a time so let us define some convenient notation. Given elements $x_1, \ldots, x_m$ we write $\boldsymbol{c} := \iota(\boldsymbol{x}) + R\boldsymbol{u}$ with $R \in \mathrm{Mat}_{m \times n}(\mathcal{R})$ for making commitments $c_1, \ldots, c_m$ computed as $c_i := \iota(x_i) + \sum_{j=1}^{n} r_{ij} u_j$.

The treatment of commitments using the language of modules generalizes several previous works dealing with commitments over bilinear groups, including [11,27,26,23,36]. We refer to the full paper [28] for a demonstration of how the commitment scheme can be instantiated with respectively the subgroup decision, the SXDH and the DLIN assumptions.

4 Setup

In our NIWI proofs the common reference string will contain commitment keys to commit to elements in respectively A_1 and A_2. These commitment keys specify $B_1, \iota_1, p_1, u_1, \ldots, u_{\hat{m}}$ so $\iota_1 \circ p_1$ is the identity map and $B_2, \iota_2, p_2, v_1, \ldots, v_{\hat{n}}$ so $\iota_2 \circ p_2$ is the identity map. In addition, the common reference string will also specify a third $\mathcal{R}$-module B_T together with $\mathcal{R}$-linear maps $\iota_T : A_T \to B_T$ and $p_T : B_T \to A_T$ so $\iota_T \circ p_T$ is the identity map. There will be a bilinear map $F : B_1 \times B_2 \to B_T$ as well. We require that the maps are commutative. We refer to Figure 4 for an overview of the modules and the maps.

For notational convenience, let us define for $\boldsymbol{x} \in B_1^n, \boldsymbol{y} \in B_2^n$ that

$$\boldsymbol{x} \bullet \boldsymbol{y} = \sum_{i=1}^{n} F(x_i, y_i).$$

The final part of the common reference string is a set of matrices $H_1, \ldots, H_\eta \in \mathrm{Mat}_{\hat{m} \times \hat{n}}(\mathcal{R})$ that all satisfy $\boldsymbol{u} \bullet H_i \boldsymbol{v} = 0$.[5]

[3] In the full paper [28], we also consider the case where $\iota \circ p$ is not the identity. In particular, in the instantiation based on the subgroup decision problem, $\iota \circ p$ is the projection on the order $\mathbf{p}$ subgroup of G.

[4] The map p is not efficiently computable. However, one can imagine scenarios where a secret key will make p efficiently computable making the commitment scheme a cryptosystem with p being the decryption operation.

[5] The number of matrices $H_1, \ldots, H_\eta$ depends on the concrete setting. In many cases, we need no matrices at all and we have $\eta = 0$, but there are also cases where they are needed.

$$\begin{array}{ccccc} A_1 & \times & A_2 & \underset{f}{\rightarrow} & A_T \\ \iota_1 \downarrow\uparrow p_1 & & \iota_2 \downarrow\uparrow p_2 & & \iota_T \downarrow\uparrow p_T \\ B_1 & \times & B_2 & \underset{F}{\rightarrow} & B_T \end{array}$$

$$\forall x \in A_1 \,\forall y \in A_2 : \; F(\iota_1(x), \iota_2(y)) = \iota_T(f(x,y))$$
$$\forall x \in B_1 \,\forall y \in B_2 : \; f(p_1(x), p_2(x)) = p_T(F(x,y))$$

Fig. 4. Modules and maps between them

There will be two different types of settings of interest to us.

Soundness setting: In the soundness setting, we require that the commitment keys are binding so we have $p_1(\boldsymbol{u}) = \boldsymbol{0}$ and $p_2(\boldsymbol{v}) = \boldsymbol{0}$.

Witness-indistinguishability setting: In the witness-indistinguishability setting we have hiding commitment keys, so $\iota_1(G_1) \subseteq \langle u_1, \ldots, u_{\widehat{m}} \rangle$ and $\iota_2(G_2) \subseteq \langle v_1, \ldots, v_{\widehat{n}} \rangle$. We also require that $H_1, \ldots, H_\eta$ generate the R-module of all matrices H so $\boldsymbol{u} \bullet H\boldsymbol{v} = 0$. As we will see in the next section, these matrices play a role as randomizers in the witness-indistinguishability proof.

Computational indistinguishability: The (only) computational assumption this paper is based on is that the two settings can be set up in a computationally indistinguishable way. The instantiations show that there are many ways to get such computationally indistinguishable soundness and witness-indistinguishability setups.

All three instantiations based on the subgroup decision, the SXDH and the DLIN assumptions enable us to make this kind of setup, see the full paper [28] for details.

5 Proving That Committed Values Satisfy a Quadratic Equation

Recall that in our setting, a quadratic equation looks like the following:

$$\boldsymbol{a} \cdot \boldsymbol{y} + \boldsymbol{x} \cdot \boldsymbol{b} + \boldsymbol{x} \cdot \Gamma \boldsymbol{y} = t, \tag{1}$$

with constants $\boldsymbol{a} \in A_1^n, \boldsymbol{b} \in A_2^m, \Gamma \in \mathrm{Mat}_{m \times n}(\mathcal{R}), t \in A_T$. We will first consider the case of a single quadratic equation of the above form. The first step in our NIWI proof will be to commit to all the variables $\boldsymbol{x}, \boldsymbol{y}$. The commitments are of the form

$$\boldsymbol{c} = \iota_1(\boldsymbol{x}) + R\boldsymbol{u} \quad , \quad \boldsymbol{d} = \iota_2(\boldsymbol{y}) + S\boldsymbol{v}, \tag{2}$$

with $R \in \mathrm{Mat}_{m \times \widehat{m}}(\mathcal{R}), S \in \mathrm{Mat}_{n \times \widehat{n}}(\mathcal{R})$. The prover's task is to convince the verifier that the commitments contain $\boldsymbol{x} \in A_1^m, \boldsymbol{y} \in A_2^n$ that satisfy the quadratic equation. (Note that for all equations we will use these same commitments.)

Intuition. Before giving the proof let us give some intuition. In the previous sections, we have carefully set up our commitments so that the commitments themselves also "behave" like the values being committed to: they also belong to modules (the B modules) equipped with a bilinear map (the map F, also implicitly used in the $\bullet$ operation). Given that we have done this, a natural idea is to take the quadratic equation (1), and "plug in" the commitments (2) in place of the variables; let us evaluate:

$$\iota_1(\boldsymbol{a}) \bullet \boldsymbol{d} + \boldsymbol{c} \bullet \iota_2(\boldsymbol{b}) + \boldsymbol{c} \bullet \Gamma \boldsymbol{d}.$$

After some computations, where we expand the commitments (2), make use of the bilinearity of $\bullet$, and rearrange terms (the details can be found in the proof of Theorem 1 in the full paper [28]) we get

$$\Big(\iota_1(\boldsymbol{a}) \bullet \iota_2(\boldsymbol{y}) + \iota_1(\boldsymbol{x}) \bullet \iota_2(\boldsymbol{b}) + \iota_1(\boldsymbol{x}) \bullet \Gamma\iota_2(\boldsymbol{y})\Big)$$
$$+\iota_1(\boldsymbol{a}) \bullet S\boldsymbol{v} + R\boldsymbol{u} \bullet \iota_2(\boldsymbol{b}) + \iota_1(\boldsymbol{x}) \bullet \Gamma S\boldsymbol{v} + R\boldsymbol{u} \bullet \Gamma\iota_2(\boldsymbol{y}) + R\boldsymbol{u} \bullet \Gamma S\boldsymbol{v}.$$

By the commutative properties of the maps, the first group of three terms is equal to $\iota_T(t)$, if Equation 1 holds. Looking at the remaining terms, note that the verifier knows $\boldsymbol{u}$ and $\boldsymbol{v}$. Using the fact that bilinearity implies that for any $\boldsymbol{x}, \boldsymbol{y}$ we have $\boldsymbol{x} \bullet \Gamma\boldsymbol{y} = \Gamma^\top \boldsymbol{x} \bullet \boldsymbol{y}$, we can sort the remaining terms so that they match either $\boldsymbol{u}$ or $\boldsymbol{v}$ to get (again see the proof of Theorem 1 in the full paper for details)

$$\iota_T(t) + \boldsymbol{u} \bullet \Big(R^\top\iota_2(\boldsymbol{b}) + R^\top\Gamma\iota_2(\boldsymbol{y}) + R^\top\Gamma S\boldsymbol{v}\Big) + \Big(S^\top\iota_1(\boldsymbol{a}) + S^\top\Gamma^\top\iota_1(\boldsymbol{x})\Big) \bullet \boldsymbol{v}. \quad (3)$$

Now, for sake of intuition, let us make some simplifying assumptions: Let's assume that we're working in a symmetric case where $A_1 = A_2$, and $B_1 = B_2$, and therefore $\boldsymbol{u} = \boldsymbol{v}$ and, so, the above equation can be simplified further to get:

$$\iota_T(t) + \boldsymbol{u} \bullet \Big(R^\top\iota_2(\boldsymbol{b}) + R^\top\Gamma\iota_2(\boldsymbol{y}) + R^\top\Gamma S\boldsymbol{u} + S^\top\iota_1(\boldsymbol{a}) + S^\top\Gamma^\top\iota_1(\boldsymbol{x})\Big).$$

Now, suppose the prover gives to the verifier as his proof $\;= \Big(R^\top\iota_2(\boldsymbol{b}) + R^\top\Gamma\iota_2(\boldsymbol{y}) + S^\top\iota_1(\boldsymbol{a}) + S^\top\Gamma^\top\iota_1(\boldsymbol{x})\Big)$. The verifier would then check that the following *verification equation* holds:

$$\iota_1(\boldsymbol{a}) \bullet \boldsymbol{d} + \boldsymbol{c} \bullet \iota_2(\boldsymbol{b}) + \boldsymbol{c} \bullet \Gamma\boldsymbol{d} = \iota_T(t) + \boldsymbol{u} \bullet \quad .$$

It is easy to see that this proof would be convincing in the soundness setting, because we have that $p_1(\boldsymbol{u}) = \boldsymbol{0}$. Then the verifier would know (but not be able to compute) that by applying the maps p_1, p_2, p_T we get

$$\boldsymbol{a} \bullet p_2(\boldsymbol{d}) + p_1(\boldsymbol{c}) \bullet \boldsymbol{b} + p_1(\boldsymbol{c}) \bullet \Gamma p_2(\boldsymbol{d}) = t + p_1(\boldsymbol{u}) \bullet p_2(\quad) = t.$$

This gives us soundness, since $\boldsymbol{x} := p_1(\boldsymbol{c})$ and $\boldsymbol{y} := p_2(\boldsymbol{d})$ satisfy the equations.

The remaining problem is to get witness-indistinguishability. Recall that in the witness-indistinguishability setting, the commitments are perfectly hiding. Therefore, in the verification equation, nothing except for has any information about $\boldsymbol{x}$ and $\boldsymbol{y}$ except for the information that can be inferred from the quadratic equation itself. So, let's consider two cases:

1. Suppose that is the unique value so that the verification equation is valid. In this case, we trivially have witness indistinguishability, since this means that all witnesses would lead to the same value for .
2. The simple case above might seem too good to be true, but let's see what it means if it isn't true. If two values and $'$ both satisfy the verification equation, then just subtracting the equations shows that $\boldsymbol{u} \bullet (\quad - \quad ') = 0$. On the other hand, recall that in the witness indistinguishability setting, the $\boldsymbol{u}$ vectors generate the entire space where or $'$ live, and furthermore we know that the matrices $H_1, \ldots, H_\eta$ generate all H such that $\boldsymbol{u} \bullet H\boldsymbol{u} = 0$. Therefore, let's choose $r_1, \ldots, r_\eta$ at random, and consider the distribution $'' = \quad + \sum_{i=1}^{\eta} r_i H_i \boldsymbol{u}$. We thus obtain the same distribution on $''$ regardless of what we started from, and such that $''$ always satisfies the verification equation.

Thus, for the symmetric case we obtain a witness indistinguishable proof system. For the general non-symmetric case, instead of having just for the $\boldsymbol{u}$ part of Equation 3, we would also have a proof for the $\boldsymbol{v}$ part. In this case, we would also have to make sure that this split does not reveal any information about the witness. What we will do is to randomize the proofs such that they get a uniform distribution on all , that satisfy the verification equation. If we pick $T \leftarrow \mathrm{Mat}_{\widehat{n} \times \widehat{m}}(\mathcal{R})$ at random we have that $\quad + T\boldsymbol{u}$ completely randomizes . The part we add in can be "subtracted" from by observing that

$$\iota_T(t) + \boldsymbol{u} \bullet \quad + \quad \bullet \boldsymbol{v} = \iota_T(t) + \boldsymbol{u} \bullet \Big(\quad - T^\top \boldsymbol{v} \Big) + \Big(\quad + T\boldsymbol{u} \Big) \bullet \boldsymbol{v}.$$

This leads to a unique distribution of proofs for the general non-symmetric case as well.

Having explained the intuition behind the proof system, we proceed to a formal description and proof of security properties.

Proof: Pick $T \leftarrow \mathrm{Mat}_{\widehat{n} \times \widehat{m}}(\mathcal{R}), r_1, \ldots, r_\eta \leftarrow \mathcal{R}$ at random. Compute

$$:= R^\top \iota_2(\boldsymbol{b}) + R^\top \varGamma \iota_2(\boldsymbol{y}) + R^\top \varGamma S \boldsymbol{v} - T^\top \boldsymbol{v} + \sum_{i=1}^{\eta} r_i H_i \boldsymbol{v}$$

$$:= S^\top \iota_1(\boldsymbol{a}) + S^\top \varGamma^\top \iota_1(\boldsymbol{x}) + T\boldsymbol{u}$$

and return the proof (,).

Verification: Return 1 if and only if

$$\iota_1(\boldsymbol{a}) \bullet \boldsymbol{d} + \boldsymbol{c} \bullet \iota_2(\boldsymbol{b}) + \boldsymbol{c} \bullet \varGamma \boldsymbol{d} = \iota_T(t) + \boldsymbol{u} \bullet \quad + \quad \bullet \boldsymbol{v}.$$

Perfect completeness of our NIWI proof will follow from the following theorem no matter whether we are in the soundness setting or the witness-indistinguishability setting. We refer to the full paper [28] for the proof.

Theorem 1. *Given* $\boldsymbol{x} \in A_1^m, \boldsymbol{y} \in A_2^n, R \in \mathrm{Mat}_{m \times \widehat{m}}(\mathcal{R}), S \in \mathrm{Mat}_{n \times \widehat{n}}(\mathcal{R})$ *satisfying*

$$\boldsymbol{c} = \iota_1(\boldsymbol{x}) + R\boldsymbol{u} \quad , \quad \boldsymbol{d} = \iota_2(\boldsymbol{y}) + S\boldsymbol{v} \quad , \quad \boldsymbol{a} \cdot \boldsymbol{y} + \boldsymbol{x} \cdot \boldsymbol{b} + \boldsymbol{x} \cdot \varGamma \boldsymbol{y} = t,$$

we have for all choices of $T, r_1, \ldots, r_\eta$ *that the proofs* , *constructed as above will be accepted.*

Perfect soundness of our NIWI proof follows from the following theorem.

Theorem 2. *In the soundness setting, where we have* $p_1(\boldsymbol{u}) = \mathbf{0}$ *and* $p_2(\boldsymbol{v}) = \mathbf{0}$*, a valid proof implies* $\boldsymbol{a} \cdot p_2(\boldsymbol{d}) + p_1(\boldsymbol{c}) \cdot \boldsymbol{b} + p_1(\boldsymbol{c}) \cdot \Gamma p_2(\boldsymbol{d}) = t$*.*

Proof. An acceptable proof , satisfies $\iota(a) \bullet \boldsymbol{d} + \boldsymbol{c} \bullet \iota_2(\boldsymbol{b}) + \boldsymbol{c} \bullet \Gamma \boldsymbol{d} = \iota_T(t) + \boldsymbol{u} \bullet \; + \; \bullet \boldsymbol{v}$. The commutative property of the linear and bilinear maps gives us

$$\begin{aligned} & p_1(\iota_1(\boldsymbol{a})) \cdot p_2(\boldsymbol{d}) + p_1(\boldsymbol{c}) \cdot p_2(\iota_2(\boldsymbol{b})) + p_1(\boldsymbol{c}) \cdot \Gamma p_2(\boldsymbol{d}) \\ &= p_T(\iota_T(t)) + p_1(\boldsymbol{u}) \cdot p_2(\quad) + p_1(\quad) \cdot p_2(\boldsymbol{v}) = p_T(\iota_T(t)). \end{aligned}$$

□

Composable witness-indistinguishability follows from the following theorem, which we prove in the full paper [28].

Theorem 3. *In the witness-indistinguishable setting where* $\iota_1(G_1) \subseteq \langle u_1, \ldots, u_{\widehat{m}} \rangle$*,* $\iota_2(G_2) \subseteq \langle v_1, \ldots, v_{\widehat{n}} \rangle$ *and* $H_1, \ldots, H_\eta$ *generate all matrices* H *so* $\boldsymbol{u} \bullet H\boldsymbol{v} = 0$*, all satisfying witnesses* $\boldsymbol{x}, \boldsymbol{y}, R, S$ *yield proofs* $\in \langle v_1, \ldots, v_{\widehat{n}} \rangle^{\widehat{m}}$ *and* $\in \langle u_1, \ldots, u_{\widehat{m}} \rangle^{\widehat{n}}$ *that are uniformly distributed conditioned on the verification equation* $\iota_1(\boldsymbol{a}) \bullet \boldsymbol{d} + \boldsymbol{c} \bullet \iota_2(\boldsymbol{b}) + \boldsymbol{c} \bullet \Gamma \boldsymbol{d} = \iota_T(t) + \boldsymbol{u} \bullet \; + \; \bullet \boldsymbol{v}$*.*

6 NIWI Proof for Satisfiability of a Set of Quadratic Equations

We will now give the full composable NIWI proof for satisfiability of a set of quadratic equations in a module with a bilinear map. The cryptographic assumption we make is that the common reference string is created by one of two algorithms K or S and that their outputs are computationally indistinguishable. The first algorithm outputs a common reference string that specifies a soundness setting, whereas the second algorithm outputs a common reference string that specifies a witness-indistinguishability setting.

Setup: $gk := (\mathcal{R}, A_1, A_2, A_T, f) \leftarrow \mathcal{G}(1^k)$.
Soundness string:
$\sigma := (B_1, B_2, B_T, F, \iota_1, p_1, \iota_2, p_2, \iota_T, p_T, \boldsymbol{u}, \boldsymbol{v}, H_1, \ldots, H_\eta) \leftarrow K(gk)$.
Witness-Indistinguishability String:
$\sigma := (B_1, B_2, B_T, F, \iota_1, p_1, \iota_2, p_2, \iota_T, p_T, \boldsymbol{u}, \boldsymbol{v}, H_1, \ldots, H_\eta) \leftarrow S(gk)$.
Proof: The input consists of gk, σ, a list of quadratic equations $\{(\boldsymbol{a}_i, \boldsymbol{b}_i, \Gamma_i, t_i)\}_{i=1}^N$ and a satisfying witness $\boldsymbol{x} \in A_1^m, \boldsymbol{y} \in A_2^n$.

Pick at random $R \leftarrow \text{Mat}_{m \times \widehat{m}}(\mathcal{R})$ and $S \leftarrow \text{Mat}_{n \times \widehat{n}}(\mathcal{R})$ and commit to all the variables as $\boldsymbol{c} := \boldsymbol{x} + R\boldsymbol{u}$ and $\boldsymbol{d} := \boldsymbol{y} + S\boldsymbol{v}$.

For each equation $(\boldsymbol{a}_i, \boldsymbol{b}_i, \Gamma_i, t_i)$ make a proof as described in Section 5. In other words, pick $T_i \leftarrow \text{Mat}_{\widehat{n} \times \widehat{m}}(\mathcal{R})$ and $r_{i1}, \ldots, r_{i\eta} \leftarrow \mathcal{R}$ compute

$$\begin{aligned} {}_i &:= R^\top \iota_2(\boldsymbol{b}_i) + R^\top \Gamma \iota_2(\boldsymbol{y}) + R^\top \Gamma S \boldsymbol{v} - T_i^\top \boldsymbol{v} + \sum_{j=1}^{\eta} r_{ij} H_j \boldsymbol{v} \\ {}_i &:= S^\top \iota_1(\boldsymbol{a}_i) + S^\top \Gamma^\top \iota_1(\boldsymbol{x}) + T_i \boldsymbol{u}. \end{aligned}$$

Output the proof $(\boldsymbol{c}, \boldsymbol{d}, \{(\;_i, \;_i)\}_{i=1}^N)$.

Verification: The input is $gk, \sigma, \{(\boldsymbol{a}_i, \boldsymbol{b}_i, \Gamma_i, t_i)\}_{i=1}^N$ and the proof $(\boldsymbol{c}, \boldsymbol{d}, \{(\ _i,\ _i)\})$. For each equation check

$$\iota_1(\boldsymbol{a}_i) \bullet \boldsymbol{d} + \boldsymbol{c} \bullet \iota_2(\boldsymbol{b}_i) + \boldsymbol{c} \bullet \Gamma_i \boldsymbol{d} = \iota_T(t_i) + \boldsymbol{u} \bullet \ _i + \ _i \bullet \boldsymbol{v}.$$

Output 1 if all the checks pass, else output 0.

The construction gives us a NIWI proof. We prove the following theorem in the full paper [28].

Theorem 4. *The protocol given above is a NIWI proof for satisfiability of a set of quadratic equations with perfect completeness, perfect soundness and composable witness-indistinguishability.*

Proof of knowledge. We observe that if K outputs an additional secret piece of information ξ that makes it possible to efficiently compute p_1 and p_2, then it is straightforward to compute the witness $\boldsymbol{x} = p_1(\boldsymbol{c})$ and $\boldsymbol{y} = p_2(\boldsymbol{d})$, so the proof is a perfect proof of knowledge.

Proof size. The size of the common reference string is $\widehat{m}$ elements in B_1 and $\widehat{n}$ elements in B_2 in addition to the description of the modules, the maps and $H_1, \ldots, H_\eta$. The size of the proof is $m + N\widehat{n}$ elements in B_1 and $n + N\widehat{m}$ elements in B_2.

Typically, $\widehat{m}$ and $\widehat{n}$ will be small, giving us a proof size that is $O(m + n + N)$ elements in B_1 and B_2. The proof size may thus be smaller than the description of the statement, which can be of size up to Nn elements in A_1, Nm elements in A_2, Nmn elements in $\mathcal{R}$ and N elements in A_T.

6.1 NIWI Proofs for Bilinear Groups

We will now outline the strategy for making NIWI proofs for satisfiability of a set of quadratic equations over bilinear groups. As we described in Section 2, there are four different types of equations, corresponding to the following four combinations of $\mathbb{Z}_{\mathbf{n}}$-modules:

Pairing product equations: $A_1 = G_1, A_2 = G_2, A_T = G_T, f(\mathcal{X}, \mathcal{Y}) = e(\mathcal{X}, \mathcal{Y})$.
Multi-scalar multiplication in G_1**:** $A_1 = G_1, A_2 = \mathbb{Z}_{\mathbf{n}}, A_T = G_1, f(\mathcal{X}, y) = y\mathcal{X}$.
Multi-scalar multiplication in G_2**:** $A_1 = \mathbb{Z}_{\mathbf{n}}, A_2 = G_2, A_T = G_T, f(x, \mathcal{Y}) = x\mathcal{Y}$.
Quadratic equations in $\mathbb{Z}_{\mathbf{n}}$**:** $A_1 = \mathbb{Z}_{\mathbf{n}}, A_2 = \mathbb{Z}_{\mathbf{n}}, A_T = \mathbb{Z}_{\mathbf{n}}, f(x, y) = xy \bmod \mathbf{n}$.

The common reference string will specify commitment schemes to respectively scalars and group elements. We first commit to all the variables and then make the NIWI proofs that correspond to the types of equations that we are looking at. It is important that we use the same commitment schemes and commitments for all equations, i.e., for instance we only commit to a scalar x once and we use the same commitment in the proof whether the equation x is involved in is a multi-scalar multiplication in G_2 or a quadratic equations in $\mathbb{Z}_{\mathbf{n}}$. The use of the same commitment in all the equations is necessary to ensure a consistent choice of x throughout the proof. As a consequence of this we use the same module B_1' to commit to x in both multi-scalar multiplication in G_2 and quadratic equations in $\mathbb{Z}_{\mathbf{n}}$. We therefore end up with at most four different modules B_1, B_1', B_2, B_2' to commit to respectively $\mathcal{X}, x, \mathcal{Y}, y$ variables. We give the full construction of efficient NIWI proofs for the three instantiations based on subgroup decision, SXDH and DLIN respectively in the full paper [28].

7 Zero-Knowledge

We will show that in many cases it is possible to make zero-knowledge proofs for satisfiability of quadratic equations. An obvious strategy is to use our NIWI proofs directly, however, such proofs may not be zero-knowledge because the zero-knowledge simulator may not be able to compute any witness for satisfiability of the equations. It turns out that the strategy is better than it seems at first sight, because we will often be able to modify the set of quadratic equations into an equivalent set of quadratic equations where a witness can be found.

We consider first the case where $A_1 = \mathcal{R}, A_2 = A_T, f(r, y) = ry$ and where S outputs an extra piece of information τ that makes it possible to trapdoor open the commitments in B_1. More precisely, τ permits the computation of $\boldsymbol{s} \in \mathcal{R}^{\widehat{m}}$ so $\iota_1(1) = \iota_1(0) + \boldsymbol{s}^\top \boldsymbol{u}$. We remark that this is a common case; in bilinear groups both multi-scalar multiplication equations in G_1, G_2 and quadratic equations in $\mathbb{Z}_{\mathbf{n}}$ have this structure.

Define $c = \iota_1(1)$ to be a commitment to $\phi = 1$. Let us rewrite the equations in the statement as

$$\boldsymbol{a}_i \cdot y + f(-\phi, t_i) + \boldsymbol{x} \cdot \boldsymbol{b}_i + \boldsymbol{x} \cdot \Gamma \boldsymbol{y} = 0.$$

We have introduced a new variable ϕ and if we choose all of our variables in these modified equations to be 0 then we have a satisfying witness. In the simulation, we give the simulator trapdoor information that permits it to open c to 0 and we can now use the NIWI proof from Section 6.

We will now describe the NIZK proof. The setup, common reference string generation, proof and verification work as a standard NIWI proof. Here we describe the simulator.

Simulation string: Using $\iota_1(1) = \iota_1(0) + \sum_{i=1}^{\widehat{m}} s_i u_i$ the simulation string is $(\sigma, \tau) := ((B_1, B_2, B_T, F, \iota_1, p_1, \iota_2, p_2, \iota_T, p_T, \boldsymbol{u}, \boldsymbol{v}), \boldsymbol{s}, H_1, \ldots, H_\eta) \leftarrow S_1(gk)$.

Simulated proof: The input consists of gk, σ, a list of quadratic equations $\{(\boldsymbol{a}_i, \boldsymbol{b}_i, \Gamma_i, t_i)\}_{i=1}^N$ and a satisfying witness $\boldsymbol{x}, \boldsymbol{y}$.

Rewrite the equations as $\boldsymbol{a}_i \cdot \boldsymbol{y} + \boldsymbol{x} \cdot \boldsymbol{b}_i + f(\phi, -t_i) + \boldsymbol{x} \cdot \Gamma_i \boldsymbol{y} = 0$. Define $\boldsymbol{x} := \mathbf{0}, \boldsymbol{y} := \mathbf{0}$ and $\phi = 0$ to get a witness that satisfies all equations.

Pick at random $R \leftarrow \mathrm{Mat}_{m \times \widehat{m}}(\mathcal{R})$ and $S \leftarrow \mathrm{Mat}_{n \times \widehat{n}}(\mathcal{R})$ and commit to all the variables as $\boldsymbol{c} := \mathbf{0} + R\boldsymbol{u}$ and $\boldsymbol{d} := \mathbf{0} + S\boldsymbol{v}$. We also use $c := \iota_1(1) = \iota_1(0) + \sum_{i=1}^{\widehat{m}} s_i u_i$ and append it to $\boldsymbol{c}$.

For each modified equation $(\boldsymbol{a}_i, \boldsymbol{b}_i, -t_i, \Gamma_i, 0)$ make a proof as described in Section 5. Return the simulated proof $\{(\boldsymbol{c}, \boldsymbol{d},\ \ _i,\ \ _i)\}_{i=1}^N$.

We prove in the full paper [28] that this construction gives us a perfect NIZK proof.

Theorem 5. *The NIWI proof from Section 6 with the simulator described above is a composable NIZK proof for satisfiability of pairing product equations with perfect completeness, soundness and composable zero-knowledge, when $A_1 = \mathcal{R}$ and the commitment in B_1 can be trapdoor opened.*

7.1 NIZK Proofs for Bilinear Groups

Let us return to the four types of quadratic equations given in Figure 1. If we set up the common reference string such that we can trapdoor open respectively $\iota_1'(1)$ and $\iota_2'(1)$

to $0 \in \mathbb{Z}_n$ then multi-scalar multiplication equations and quadratic equations in $\mathbb{Z}_\mathbf{n}$ are of the form for which we can give zero-knowledge proofs (at no additional cost).

In the case of pairing product equations we do not know how to get zero-knowledge, since even with the trapdoors we may not be able to compute a satisfiability witness. We do observe though that in the special case, where all $t_T = 1$ the choice of $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{O}}$ is a satisfactory witness. Since we also use $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{O}}$ in the other zero-knowledge proofs, the simulator can use this witness and give a NIWI proof. In the special case where all $t_T = 1$ we can therefore make NIZK proofs for satisfiability of the set of pairing product equations.

Next, let us look at the case where we have a pairing product equation with $t_T = \prod_{i=1}^n e(\mathcal{P}_i, \mathcal{Q}_i)$ for some known $\mathcal{P}_i, \mathcal{Q}_i$. In this case, we can add linear equations $\mathcal{Z}_i = \mathcal{P}_i$ to the set of multi-scalar multiplication equations in G_1. We already know that such equations have zero-knowledge proofs. We can now rewrite the pairing product equation as $(\boldsymbol{\mathcal{A}} \cdot \boldsymbol{\mathcal{Y}})(\boldsymbol{\mathcal{X}} \cdot \boldsymbol{\mathcal{B}})(\boldsymbol{\mathcal{Z}} \cdot \boldsymbol{\mathcal{Q}})(\boldsymbol{\mathcal{X}} \cdot \Gamma \boldsymbol{\mathcal{Y}}) = 1$. We can therefore also make zero-knowledge proofs if all the pairing product equations have t_T of the form $t_T = \prod_{i=1}^n e(\mathcal{P}_i, \mathcal{Q}_i)$ for some known $\mathcal{P}_i, \mathcal{Q}_i$.

The case of pairing product equations points to a couple of differences between witness-indistinguishable proofs and zero-knowledge proofs using our techniques. NIWI proofs can handle any target t_T, whereas zero-knowledge proofs can only handle special types of target t_T. Furthermore, if $t_T \neq 1$ the size of the NIWI proof for this equation is constant, whereas the NIZK proof for the same equation may be larger.

Acknowledgements

We gratefully acknowledge Brent Waters for a number of helpful ideas, comments, and conversations related to this work. In particular, our module-based approach can be seen as formalizing part of the intuition expressed by Waters that the Decisional Linear Assumption, Subgroup Decision Assumption in composite-order groups, and SXDH can typically be exchanged for one another. (We were inspired by previously such connections made by [26,36].) It would be interesting to see if this intuition can be made formal in other settings, such as Traitor Tracing [12] or Searchable Encryption [13]. We also thank Dan Boneh for his encouragement and for suggesting using our techniques to get fair exchange.

References

1. Ballard, L., Green, M. de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417 (2005), http://eprint.iacr.org/2005/417
2. Barreto, P.: The pairing-based crypto lounge (2006), http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html
3. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Non-interactive anonymous credentials. In: PACS 2000. LNCS (2008), http://eprint.iacr.org/2007/384
4. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC, pp. 103–112 (1988)

5. Boneh, D.: Personal communication (2006)
6. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
8. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
9. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
10. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
11. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
12. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
13. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
14. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007), http://www.cs.stanford.edu/~xb/pkc07/
15. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007)
16. Damgård, I.: Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 341–355. Springer, Heidelberg (1993)
17. De Santis, A., Di Crescenzo, G., Persiano, G.: Randomness-optimal characterization of two NP proof systems. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 179–193. Springer, Heidelberg (2002)
18. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM Journal of Computing 30(2), 391–437 (2000)
19. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs under general assumptions. SIAM Journal of Computing 29(1), 1–28 (1999)
20. Galbraith, S.D., Rotger, V.: Easy decision Diffie-Hellman groups. London Mathematical Society Journal of Computation and Mathematics 7, 201–218 (2004)
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proofs. SIAM Journal of Computing 18(1), 186–208 (1989)
22. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS, pp. 89–98 (2006)
23. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 444–459. Springer, Heidelberg (2006), http://www.brics.dk/~jg/NIZKGroupSignFull.pdf
24. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007), http://www.brics.dk/~jg/CertiSignFull.pdf
25. Groth, J., Lu, S.: A non-interactive shuffle with pairing based verifiability. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67. Springer, Heidelberg (2007)

26. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
27. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero-knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
28. Groth, J. Sahai, A.: Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155 (2007), http://eprint.iacr.org/2007/155
29. Kilian, J., Petrank, E.: An efficient noninteractive zero-knowledge proof system for NP with general assumptions. Journal of Cryptology 11(1), 1–27 (1998)
30. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: PODC, pp. 12–19 (2003)
31. Paterson, K.G.: Cryptography from pairings. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, vol. 317, pp. 215–251. Cambridge University Press, Cambridge (2005)
32. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
33. Scott, M.: Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive, Report 2002/164 (2002), http://eprint.iacr.org/2002/164
34. Verheul, E.R.: Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. Journal of Cryptology 17(4), 277–296 (2004)
35. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
36. Waters, B.: New techniques for slightly 2-homomorphic encryption, Manuscript (2006)

Zero-Knowledge Sets with Short Proofs*

Dario Catalano[1], Dario Fiore[1], and Mariagrazia Messina[2,**]

[1] Dipartimento di Matematica ed Informatica – Università di Catania, Italy
{catalano,fiore}@dmi.unict.it
[2] Microsoft Italia
mariame@microsoft.com

Abstract. Zero Knowledge Sets, introduced by Micali, Rabin and Kilian in [17], allow a prover to commit to a secret set S in a way such that it can later prove, non interactively, statements of the form $x \in S$ (or $x \notin S$), without revealing any further information (on top of what explicitly revealed by the inclusion/exclusion statements above) on S, not even its size. Later, Chase *et al.* [5] abstracted away the Micali, Rabin and Kilian's construction by introducing an elegant new variant of commitments that they called (trapdoor) mercurial commitments. Using this primitive, it was shown in [5,4] how to construct zero knowledge sets from a variety of assumptions (both general and number theoretic).

In this paper we introduce the notion of trapdoor q-mercurial commitments (qTMCs), a notion of mercurial commitment that allows the sender to commit to an *ordered* sequence of exactly q messages, rather than to a single one. Following [17,5] we show how to construct ZKS from qTMCs and collision resistant hash functions.

Then, we present an efficient realization of qTMCs that is secure under the so called Strong Diffie Hellman assumption, a number theoretic conjecture recently introduced by Boneh and Boyen in [3]. Using our scheme as basic building block, we obtain a construction of ZKS that allows for proofs that are much shorter with respect to the best previously known implementations. In particular, for an appropriate choice of the parameters, our proofs are up to 33% shorter for the case of proofs of membership, and up to 73% shorter for the case of proofs of non membership.

1 Introduction

Imagine some party P wants to commit to a set S, in a way such that any other party V can "access" S in a limited but reliable manner. By limited here we mean that V is given indirect access to S, in the sense that she is allowed to ask only questions of the form "is x in S?". P answers such questions by providing publicly verifiable proofs for the statements $x \in S$ (or $x \notin S$). Such proofs should be reliable in the sense that a cheating P should not be able to convince V that some x is in the set while is not (or viceversa). At the same time, they should be "discreet" enough not to reveal anything beyond their validity.

* The full version of this paper is available at http://www.dmi.unict.it/~fiore
** Work entirely done while student at University of Catania.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 433–450, 2008.

The notion of Zero Knowledge Sets (ZKS) was recently introduced by Micali, Rabin and Kilian [17] to address exactly this problem. Informally, ZKS allow a prover P to commit to an arbitrary (but finite) set S in a way such that P can later prove statements of the form $x \in S$ or $x \notin S$ without revealing any significant information about S (not even its size!). As already pointed out in [17], the notion of zero knowledge sets can be easily extended to encompass the more general notion of elementary databases (EDB). In a nutshell, an elementary database is a set S with the additional property that each $x \in S$ comes with an associated value $S(x)$. In the following we will refer to ZKS to include zero knowledge EDB as well.

The solution by Micali *et al.* is non interactive and works in the so called *shared random string* model (i.e. where a random string, built by some trusted third party, is made available to all participants) building upon a very clever utilization of a simple commitment scheme, originally proposed by Pedersen [20].

Commitment schemes play a central role in cryptography. Informally, they can be seen as the digital equivalent of an opaque envelope. Whatever is put inside the envelope remains secret until the latter is opened (hiding property) and whoever creates the commitment should not be able to open it with a message that is not the one originally inserted (binding property). Typically, a commitment scheme is a two phase procedure. During the first phase, the sender creates a commitment C, to some message m, using an appropriate commitment algorithm, and sends C to the receiver R. In the opening phase the sender opens C by giving R all the necessary material to (efficiently) verify that C was indeed a valid commitment to m.

Since Pedersen's commitment relies on the intractability of the discrete logarithm, so does the construction in [17]. Later, Chase *et al.* [5] abstracted away Micali *et al.*'s solution and described the exact properties a commitment scheme should possess in order to allow a similar construction. This led to an elegant new variant of commitments, that they called *mercurial commitment*.

Informally, a mercurial commitment is a commitment scheme where the binding requirement is somewhat relaxed in order to allow for two decommitment procedures: an *hard* and a *soft* one. At committing time, the sender can decide as whether to create an *hard* commitment or a *soft* one, from the message m he has in mind. Hard commitments are like standard ones, in the sense that they can be (hard or soft) opened only with respect to the message that was originally used to construct the commitment. Soft commitments, on the other hand, allow for more freedom, as they cannot be hard opened in any way, but they can be soft opened to any arbitrary message. An important requirement of mercurial commitments is that, hard and soft commitments should look alike to any polynomially bounded observer.

Using this new primitive, Chase *et al.* proved that it is possible to construct ZKS from a variety of assumptions (number theoretic or general)[1]. Their most

[1] More precisely, they require the mercurial commitment to be *trapdoor* as well. Very informally, this means that the scheme comes with a trapdoor information *tk* (normally not available to anyone) that allows to completely destroy the binding property of the commitment.

general implementation, shows that (non interactive) ZKS can be constructed, in the shared random string model, assuming non interactive zero knowledge proofs (NIZK) [2] and collision resistant hash functions [8][2]. Moreover, they showed that collision resistant hash function are necessary to construct ZKS, as they are implied by the latter. Finally, Catalano, Dodis and Visconti [4] gave a construction of (trapdoor) mercurial commitments from one way functions in the shared random string model. This result completed the picture as it showed that collision resistant hash functions are necessary and sufficient for non interactive ZKS in the shared random string model.

Our Contribution. All the constructions above, build upon the common idea of constructing an authenticated Merkle tree of depth k where each internal node is a mercurial commitment (rather than the hash) of its two children. Very informally, to prove that a given $x \in \{0,1\}^k$ belongs to the committed set S, the prover simply opens all the commitments in the path from the root to the leaf labeled by x (more details about this methodology will be given later on). Thus the length of the resulting proof is $k \cdot d$, where d denotes the length of the opening of the mercurial commitment, and k has to be chosen so that 2^k is larger than the size of any "reasonably" large set S[3]. Assuming $k = 128$ and $d = O(k)$, as it is the case for all known implementations, this often leads to very long proofs.

It is thus important to research if using the properties of specific number-theoretic problems, it is possible to devise zero knowledge sets that allow for shorter proofs. Such proofs would be desirable in all those scenarios where space or bandwidth are limited. A typical example of such a scenario is mobile internet connections, where customers pay depending on the number of blocks sent and received.

In this paper, we present a new construction of ZKS that allows for much shorter proofs, with respect to the best currently known implementation (which is the Micali *et al.* construction when implemented on certain classes of elliptic curves. From now on we will use the acronym MRK to refer to such an implementation).

Our solution relies on a new primitive that we call *trapdoor q-Mercurial Commitment* (qTMC, for short). Informally, qTMCs allow the sender to commit to a *sequence* of exactly q messages $(m_1, \ldots, m_q)$, rather than to a single one, as with standard mercurial commitments. The sender can later open the commitment with respect to any message m_i but, in order to do so successfully, he has to correctly specify the exact position held by the message in the sequence. In other

[2] It is known that one can construct NIZK under the assumption that trapdoor permutations exist or under the assumption that verifiable random functions (VRF) exist [12,18]. These two assumptions, however, are, as far as we currently know, incomparable.

[3] This is because 2^k is also an upper bound for the size of the set. Thus, to meet the requirements of ZKS it should not reveal anything about the cardinality of the set itself.

words, trapdoor q-Mercurial commitments allow to commit to *ordered* sequences of q messages.

Following [17,5], we show how to construct ZKS from qTMCs and collision resistant hash functions. This step is rather simple but very useful for our goal, as it reduces the task of realizing efficient ZKS to the task of realizing efficient qTMCs. Indeed, even though the proposed transformation allows us to use a "flat" Merkle tree (i.e. with branching factor q, rather than two), it *does not* lead, by itself, to shorter proofs.

Recall that, informally, a proof for the statement $x \in S$ (or $x \notin S$) consists of an authenticated path from the root to the leaf labeled by x. The trouble is that in all known implementations of ZKS, to verify the authenticity of a node in the path, one must know all siblings of the node. If the tree is binary, the proof contains twice as many nodes as the the depth of the tree (since each node must be accompanied by its sibling). Thus, the length of a proof being proportional to the branching factor of the tree, increasing the latter, is actually a *bad* idea in general. Indeed, suppose we want to consider sets defined over a universe of N elements. Using a binary authentication tree one gets proofs whose length is proportional to $\log_2 N(2n)$, where n is the size of the authentication information contained in each node. Using a tree with branching degree q, on the other hand, one would get proofs of size $\log_q N(qn)$, which is actually more than in previous case.

Overcoming the proofs blow-up. In this paper we propose an implementation of trapdoor q mercurial commitments that overcomes the above limitation. Our solution relies on the so called Strong Diffie Hellman assumption originally introduced by Boneh and Boyen [3] and builds upon the weakly secure digital signature given in [3]. The proposed implementation exploits the algebraic properties of the employed number theoretic primitive to produce a qTMC that allows for short openings. More precisely the size of each hard opening still depends linearly on q, but the size of each soft opening becomes constant and completely *independent* of q.

This results in ZKS that allow for much shorter proofs than MRK. Concretely, and for an appropriate choice of the parameter q, our proofs are up to 33% shorter for the case of proofs of membership, and up to 73% shorter for the case of proofs of non membership.

Zero Knowledge Sets vs Signatures. The idea of obtaining short proofs by changing the authentication procedure to deal with a "flat" authentication tree, is reminiscent of a technique originally suggested by Dwork and Naor [9], in the context of digital signature schemes. In a nutshell, the Dwork-Naor method allows to increase the branching factor of the tree without inflating the signature size. This is achieved, by, basically, authenticating each node with respect to its parent, but without providing its siblings.

Adapting this idea to work to the case of zero knowledge sets, presents several non trivial technical difficulties[4]. The main problem comes from the fact that, in

[4] It is probably instructive to mention the fact that, indeed, the Dwork Naor solution, and its improvements such as [7], *do not* work in our setting

ZKS, one has to make sure that a dishonest prover cannot construct two, both valid, proofs for the statements $x \in S$ and $x \notin S$. Such a requirement imposes limitations just not present when dealing with digital signatures [5].

Other Related work. Ostrovsky, Rackoff and Smith [19] described a construction that allows a prover to commit to a database and to provide answers that are consistent with the commitment. Their solution can handle more elaborate queries than just membership ones. Moreover they also consider the issue of adding privacy to their protocol. However their construction requires interaction (at least if one wants to avoid the use of random oracles) and requires the prover to keep a counter for the questions asked so far.

Gennaro and Micali [11] recently introduced the notion of independent zero knowledge sets. Informally, independent ZKS protocols prevent an adversary from successfully correlate her set to the one of a honest prover. Their notion of independence also implies that the resulting ZKS protocol is non-malleable and requires a new commitment scheme that is both independent and mercurial. We do not consider such an extension here.

Liskov [15] considered the problem of updating zero-knowledge databases. In [15] definitions for updatable zero knowledge databases are given, together with a construction based on verifiable random functions [18] and mercurial commitments. The construction, however, is in the random oracle model [1].

Very recently Prabhakaran and Xue [21] introduced the notion of statistically hiding sets (SHS) that is related but different than ZKS. Informally, SHS require the hiding property to hold with respect to unbounded verifiers. At the same time, however, they relax the zero knowledge requirement to allow for unbounded simulators.

Road Map. The paper is organized as follows. In section 2 we introduce the notion of trapdoor q mercurial commitments and provide the relevant definitions for zero knowledge sets. Section 3 is devoted to the construction of ZKS from trapdoor q mercurial commitments. In section 4 we show how to construct efficient qTMCs from the Strong Diffie Hellman Assumption. Efficiency considerations and comparisons with previous work are given in section 5. Finally conclusions and directions for future work are given in section 6.

2 Preliminaries

Informally, we say that a function is *negligible* if it vanishes faster than the inverse of any polynomial.

[5] For instance, the soundness requirement above, imposes that exactly one single path from the root to a leaf, should be "labelable" as x. It seems very hard (if at all possible) to achieve this, when both type of proofs (i.e. proofs of membership and proofs of non membership) allow to authenticate each node (with respect to its parent), without providing its siblings.

2.1 Trapdoor q-Mercurial Commitments

Informally, a trapdoor q-mercurial commitment (qTMC for brevity) extends the notion of (trapdoor) mercurial commitment, by allowing the sender to commit to an (ordered) sequence of q messages, rather than to a single one. More precisely, and like standard (trapdoor) mercurial commitments (see [4] for a definition of trapdoor mercurial commitments), trapdoor q-mercurial commitments allows for two different decommitting procedures. In addition to the standard opening mechanism, there is a partial opening (also referred as *tease* or soft open) algorithm that allows for some sort of equivocation. At committing stage, the sender can decide to produce a commitment in two ways. Hard commitments should be hiding in the usual sense, but should satisfy a very strong binding requirement (that we call strong binding). Informally, strong binding means that a sender S should be able to open a commitment only with respect to messages that were in the "correct" position in the sequence S originally committed to. More precisely, when opening an hard commitment for a message m, the sender is required to specify an index $i \in \{1, \ldots, q\}$, indicating the position of m in the sequence. In the case of hard commitments, the strong binding property imposes that the commitment should be successfully opened and teased to (m, i) only if m was the i-th message in the sequence S originally committed to. Soft commitments, on the other hand cannot be opened, but can be teased with respect to messages belonging to any arbitrary sequence of q messages.

More formally, a trapdoor q-mercurial commitment is defined by the following set of algorithms: ($\mathsf{qKeyGen}, \mathsf{qHCom}, \mathsf{qHOpen}$, $\mathsf{qHVer}, \mathsf{qSCom}, \mathsf{qSOpen}$, qSVer, $\mathsf{qFake}, \mathsf{qHEquiv}$, $\mathsf{qSEquiv}$).

$\mathsf{qKeyGen}(1^k, q)$ is a probabilistic algorithm that takes in input a security parameter k and the number q of committed values and outputs a pair of public/private keys (pk, tk).

$\mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ Given an ordered tuple of messages, qHCom computes a hard commitment C to $(m_1, \cdots, m_q)$ using the public key pk and returns some auxiliary information aux.

$\mathsf{qHOpen}_{pk}(m, i, \mathsf{aux})$ Let $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$, if $m = m_i$ the hard opening algorithm $\mathsf{qHOpen}_{pk}(m, i, \mathsf{aux})$ produces a hard decommitment π. The algorithm returns an error message otherwise.

$\mathsf{qHVer}_{pk}(m, i, C, \pi)$ The hard verification algorithm $\mathsf{qHVer}_{pk}(m, i, C, \pi)$ accepts (outputs 1) only if π proves that C is created to a tuple $(m_1, \cdots, m_q)$ such that $m_i = m$.

$\mathsf{qSCom}_{pk}()$ produces a soft commitment C and an auxiliary information aux. A soft commitment string C is created to no specific sequence of messages.

$\mathsf{qSOpen}_{pk}(m, i, \mathsf{flag}, \mathsf{aux})$ produces a soft decommitment τ (also known as *"tease"*) to a message m at position i. The parameter $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$ indicates if τ corresponds to either a hard commitment $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}$ $(m_1, \cdots, m_q)$ or to a soft commitment $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$. The algorithm returns an error message if C is an hard commitment and $m \neq m_i$.

$\mathsf{qSVer}_{pk}(m, i, C, \tau)$ checks if τ is a valid decommitment for C to m of index i. If it outputs 1 and τ corresponds to a hard commitment C to $(m_1, \cdots, m_q)$, then C could be hard-opened to (m, i), or rather $m = m_i$.

$\mathsf{qFake}_{pk,tk}()$ takes as input the trapdoor tk and produces a q-fake commitment C. C is not bound to any sequence $(m_1, \cdots, m_q)$. It also returns an auxiliary information aux.

$\mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathsf{aux})$ The non-adaptive hard equivocation algorithm generates a hard decommitment π for $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$ to the i-th message of $(m_1, \cdots, m_q)$. The algorithm is non adaptive in the sense that, for a given C, the sequence $(m_1, \cdots, m_q)$ has to be determined once and for all, before $\mathsf{qHEquiv}$ is executed. A q-fake commitment is very similar to a soft commitment with the additional property that it can be hard-opened.

$\mathsf{qSEquiv}_{pk,tk}(m, i, \mathsf{aux})$ generates a soft decommitment τ to m of position i using the auxiliary information produced by the qFake algorithm.

The correctness requirements for trapdoor q-Mercurial commitments are essentially the same as those for "traditional" commitment schemes. In particular we require that $\forall (m_1, \cdots, m_q) \in \mathcal{M}^q$, the following statements are false only with negligible probability.

1. if $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$:
$$\mathsf{qHVer}_{pk}(m_i, i, C, \mathsf{qHOpen}_{pk}(m_i, i, \mathsf{aux})) = 1 \quad \forall i = 1 \dots q$$
2. If $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$
$$\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux})) = 1 \quad \forall i = 1 \dots q$$
3. If $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$
$$\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{S}, \mathsf{aux})) = 1 \quad \forall i = 1 \dots q$$
4. If $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$
$$\mathsf{qHVer}_{pk}(m_i, i, C, \mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathsf{aux})) = 1$$
$$\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSEquiv}_{pk,tk}(m_i, i, \mathsf{aux})) = 1 \quad \forall i = 1 \dots q$$

Security. The security properties for a trapdoor q-mercurial commitment scheme are as follows:

– **q-Mercurial binding.** Having knowledge of pk it is computationally infeasible for an algorithm $\mathcal{A}$ to come up with C, m, i, π, m', π' such that either one of the following cases holds:
  - π is a valid hard decommitment for C to (m, i) and π' is a valid hard decommitment for C to (m', i), with $m \neq m'$. We call such case a "hard collision".

  - π is a valid hard decommitment for C to (m, i) and π' is a valid soft decommitment for C to (m', i), with $m \neq m'$. We call such case a "soft collision".

- q**-Mercurial hiding.** There exists no PPT adversary $\mathcal{A}$ that, knowing pk, can find a tuple $(m_1, \cdots, m_q) \in \mathcal{M}^q$ and an index i for which it can distinguish $(C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux}))$ from $(C', \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{S}, \mathsf{aux}'))$, where $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ and $(C', \mathsf{aux}') = \mathsf{qSCom}_{pk}()$.
- **Equivocations.** There exists no PPT adversary $\mathcal{A}$ that, knowing pk and the trapdoor tk, can win any of the following games with non-negligible probability. In such games $\mathcal{A}$ should be able to tell apart the "real" world from the corresponding "ideal" one. The games are formalized in terms of a challenger that flips a binary coin $b \in \{0, 1\}$. If $b = 0$ it gives to $\mathcal{A}$ a real commitment/decommitment tuple; if $b = 1$ it gives to $\mathcal{A}$ an ideal tuple produced using the fake algorithms.

  In the q-HHEquivocation and the q-HSEquivocation games below, $\mathcal{A}$ chooses $(m_1, \cdots, m_q) \in \mathcal{M}^q$ and receives a commitment string C. Then $\mathcal{A}$ gives an index $i \in \{1, \cdots, q\}$ to the challenger and finally it receives a hard decommitment π.

  - **q-HHEquivocation.** If $b = 0$ the challenger hands to $\mathcal{A}$ the value $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$. $\mathcal{A}$ gives i to the challenger and gets back $\pi = \mathsf{qHOpen}_{pk}(m_i, i, \mathsf{aux})$. Otherwise the challenger computes $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$, $\pi = \mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathsf{aux})$.
  - **q-HSEquivocation.** The challenger computes $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$, $\pi = \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux})$ in the case $b = 0$ or $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}(), \pi = \mathsf{qSEquiv}_{pk,tk}(m_i, i, \mathsf{aux})$ if $b = 1$.
  - **q-SSEquivocation.** If $b = 0$ the challenger generates $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$ and gives C to $\mathcal{A}$. Next, $\mathcal{A}$ chooses $m \in \mathcal{M}$ and an index $i \in \{1, \cdots, q\}$, it gives (m, i) to the challenger and receives back $\mathsf{qSOpen}_{pk}(m, i, \mathbb{S}, \mathsf{aux})$. If $b = 1$, $\mathcal{A}$ first gets $\mathsf{qFake}_{pk,tk}()$, then it chooses $m \in \mathcal{M}, i \in \{1, \cdots, q\}$, gives (m, i) to the challenger and gets back $\mathsf{qSEquiv}_{pk,tk}(m, i, \mathsf{aux})$.

  At some point $\mathcal{A}$ outputs b' as its guess for b and wins if $b' = b$.
  As for the case of trapdoor mercurial commitments (see [4]) it is easy to see that the q-mercurial hiding is implied by the q-HSEquivocation and q-SSEquivocation.

2.2 Zero-Knowledge Sets

Zero knowledge sets [17] allows one to commit to some secret set S and then to, non interactively, produce proofs of the form $x \in S$ or $x \notin S$. This is done without revealing any further information (i.e. that cannot be deduced by the statements above) about S, not even its size. Following the approach of [17], here we focus on the more general notion of zero-knowledge elementary databases (EDB), since the notion of zero-knowledge sets is a special case of zero-knowledge EDBs (see [17] for more details about this). Let $[D]$ be the set of keys associated to a database D. We assume that $[D]$ is a proper subset of $\{0,1\}^*$. If $x \in [D]$, we denote with $y = D(x)$ its associated value in the database D. If $x \notin [D]$ we let $D(x) = \perp$. An EDB system is formally defined by a triple of algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$:

– P_1, the *committer* algorithm, takes in input a database D and the common reference string CRS and outputs a public key ZPK and a secret key ZSK.
– On input the common reference string CRS, the secret key ZSK and an element x, the *prover* algorithm P_2 produces a proof π_x of either $D(x) = y$ or $D(x) = \perp$.
– The third algorithm is the *verifier* $\mathsf{V}(CRS, ZPK, x, \pi_x)$. It outputs y if $D(x) = y$, *out* if $D(x) = \perp$ or $\perp$ if the proof π_x is not valid.

The formal definition of *zero-knowledge EDB* is given in [17][6].

3 Zero Knowledge EDB from Trapdoor q-Mercurial Commitments

In this section we describe a construction of zero-knowledge EDB, from trapdoor q-mercurial commitments (defined in section 2.1), trapdoor mercurial commitments [5,4] and collision resistant hash functions. The construction is very simple, as it generalizes easily from the original [17,5] constructions. Still, it plays an important role in our quest for efficient zero knowledge sets, as it allows us to concentrate solely on the problem of realizing efficient qTMCs.

Intuitive Construction. Assume we want to commit to a database D with keys of length k. We associate each key x to a leaf in a q-ary tree of height k. Thus x can be viewed as a number representing the labeling of the leaf in q-ary encoding (see the example in Figure 1). Since the number of all possible keys is q^k, to make the committing phase efficient (i.e. polynomial in k) the tree is pruned by cutting those subtrees containing only keys of elements not in the database. The roots of such subtrees are kept in the tree (we call them the "frontier"). The internal nodes in the frontier are "filled" with soft commitments. The remaining nodes are filled as follows. Each leaf contains an hard commitment (computed using the standard trapdoor mercurial commitment scheme) of a value $n_{H(x)}$ related to $D(x)$[7]. Each internal node contains the hard q-commitment to the hashes of the values contained in its q sons. The q-commitment contained in the root of the tree is the public key of the zero-knowledge EDB.

When the prover P is asked for a proof of an element $x \in D$ (for instance such that $D(x) = y$), it proceeds as follows. It exhibits hard openings for the commitments contained in the nodes in the path from the root to the leaf x.

[6] We point out here that we will prove our construction secure with respect to a slightly different definition (with respect to the one given in [17]) in which the completeness requirement is relaxed to allow a negligible probability of error.

[7] More precisely $n_{H(x)}$ is the hash of $D(x)$ if x is in the database and 0 otherwise.

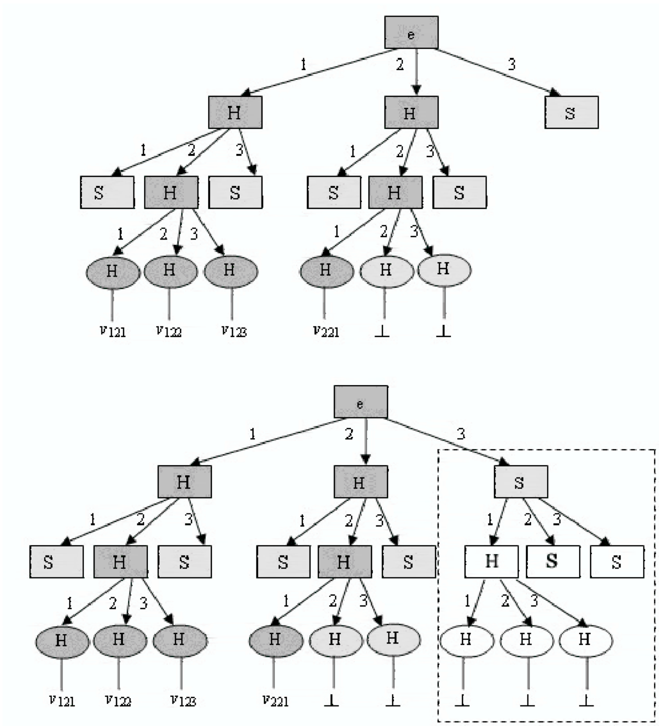


Fig. 1. A 3-ary tree of height 3 before and after a query to the database key 311. Each node of the tree contains a mercurial commitment: the label H is for hard commitments, S for the soft ones. Moreover the squares represent q-commitments, while the circles represent standard commitments. If the set of database keys is $S = \{121, 122, 123, 221\}$, the darker nodes are those belonging to a path from the root to an element in the set. The light shaded nodes are the frontier.

More precisely, for each level of the tree, it opens the hard q-commitment with respect to the position determined by the q-ary encoding of x for that level. Queries corresponding to keys x such that $D(x) = \perp$ are answered as follows. First, the prover generates the possibly missing portion of the subtree containing x. Next, it soft opens all the commitments contained in the nodes in the path from x to the root. The soft commitments stored in the frontier nodes are then teased to the values contained in its newly generated children.

It is easy to see that the completeness property follows from the completeness of the two commitment schemes used. Similarly, the binding properties of the two commitment schemes, together with the collision resistance of the underlying hash function, guarantees that (1) no hard commitment can be opened to two different values, and (2) no hard commitment can be opened to a value and then teased to a different one.

Finally the zero-knowledge property follows from the fact that both the two commitments schemes are hiding and equivocable (the fake commitments and fake openings produced by the simulator are indistinguishable from the commitments and openings produced from a real prover).

A detailed description of the construction sketched above, together with a complete security proof, is given in the full version of this paper.

4 Trapdoor q-Mercurial Commitment Based on SDH

In this section we show an efficient construction of trapdoor q-mercurial commitments $\mathcal{QC}$.

Our construction relies on the Strong Diffie-Hellman assumption (SDH for short), introduced by Boneh and Boyen in [3]. Informally, the SDH assumption in bilinear groups G_1, G_2 of prime order p states that, for every PPT algorithm $\mathcal{A}$ and for a parameter q, the following probability is negligible:

$$Pr[\mathcal{A}(g_1, g_1^x, g_1^{(x^2)}, \cdots, g_1^{(x^q)}, g_2, g_2^x) = (c, g_1^{1/(x+c)})].$$

If we suppose that $\mathcal{G}(1^k)$ is a bilinear group generator which takes in input a security parameter k, then (asymptotically) the SDH assumption holds for $\mathcal{G}$ if the probability above is negligible in k, for any q polynomial in k (see [3] for the formal definition).

The SDH assumption obviously implies the discrete logarithm assumption (i.e. if the former holds, so has to do the latter). A reduction in the other direction, however, is not known. Recently, however, Cheon [6] proved that, for many primes p, the q-Strong Diffie Hellman problem has computational complexity reduced by $O(\sqrt{q})$ with respect to that of the discrete logarithm problem (in the same group).

THE NEW SCHEME. Now we describe our proposed trapdoor q-Mercurial Commitment scheme, in terms of its component algorithms ($\mathsf{qKeyGen}, \mathsf{qHCom}$, qHOpen, qHVer, $\mathsf{qSCom}, \mathsf{qSOpen}$, $\mathsf{qSVer}, \mathsf{qFake}, \mathsf{qHEquiv}$, $\mathsf{qSEquiv}$), as described in section 2.1.

The technical construction of the proposed scheme builds upon the simulator described in the security proof of the weak signature scheme given in [3].

In what follows $\mathcal{H}$ denotes a family of collision resistant hash functions whose range is $\mathbb{Z}_p$.

$\mathsf{qKeyGen}(1^k, q)$ The key generation algorithm runs a bilinear group generator $\mathcal{G}(1^k)$ for which the SDH assumption holds [3] to get back the description of groups G_1, G_2, G_T and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$. Such groups share the same prime order p.

The description of the groups contains the group generators: $g_1 \in G_1$, $g_2 \in G_2$. The algorithm proceeds by picking a random integer $x \leftarrow \mathbb{Z}_p^*$ and it sets $A_1 = g_1^x, \cdots, A_q = g_1^{x^q}, h = g_2^x$. Next, it chooses a collision resistant hash function H from $\mathcal{H}$.

The public key is set as $PK = (g_1, A_1, \cdots, A_q, g_2, h, H)$, while the trapdoor is $TK = x$.

$\mathsf{qHCom}_{PK}(m_1, \cdots, m_q)$. The hard commitment algorithm randomly selects $\alpha, w \leftarrow \mathbb{Z}_p^*$ and computes $C_i = H(i||m_i), \forall i = 1, \cdots, q$ (the symbol $||$ denotes concatenation). Next, it defines the polynomial

$$f(z) = \prod_{i=1}^{q}(z + C_i) = \sum_{i=0}^{q}(\beta_i z^i)$$

and sets $g_1' = (\prod_{i=0}^{q} A_i^{\alpha^i \beta_i})^w = g_1^{f(\alpha x)w}$ and $g_2' = h^\alpha$. In the unlucky case that either $g_1' = 1$ or $g_2' = 1$, then one simply retries with another random α.

Thus, letting $\gamma = \alpha x$, we have $g_1' = g_1^{f(\gamma)w}$ and $g_2' = h^\alpha = g_2^\gamma$.

The commitment is (g_1', g_2'). The auxiliary information is $\mathsf{aux} = (\alpha, w, m_1, \cdots, m_q)$.

$\mathsf{qHOpen}_{PK}(m, j, \mathsf{aux})$ outputs $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$.

$\mathsf{qHVer}_{PK}(m, j, C, \pi)$ computes the $q - 1$ terms $C_i = H(i||m_i) \forall m_i \in \pi$ and $C_j = H(j||m)$. Next, it defines the polynomial $f(z) = \prod_{i=1}^{q}(z + C_i)$ and computes the β_i coefficients as above.

Checks if $g_1' = (\prod_{i=0}^{q} A_i^{\alpha^i \beta_i})^w$ and $g_2' = h^\alpha$. If both tests succeed, it outputs 1.

$\mathsf{qSCom}_{PK}()$ picks $\alpha', y \leftarrow \mathbb{Z}_p^*$ at random, sets

$$g_1' = g_1^{\alpha'}, \;\; g_2' = g_2^y$$

and outputs (g_1', g_2') and $\mathsf{aux} = (\alpha', y)$.

$\mathsf{qSOpen}_{PK}(m, j, \mathsf{flag}, \mathsf{aux})$ If $\mathsf{flag} = \mathbb{H}$ the algorithm computes $C_i = H(i||m_i)$, $\forall i = 1, \cdots, j-1, j+1, \cdots, q$, $C_j = H(j||m)$, it sets

$$f_j(z) = \frac{f(z)}{(z + C_j)} = \prod_{i=1 \wedge i \neq j}^{q}(z + C_i) = \sum_{i=0}^{q-1} \delta_i z^i$$

Next, it computes $\sigma_j = (\prod_{i=0}^{q-1} A_i^{\delta_i \alpha^i})^w = g_1^{\frac{f(\gamma)w}{\gamma + C_j}} = (g_1')^{\frac{1}{\gamma + C_j}}$. The output is σ_j.

If $\mathsf{flag} = \mathbb{S}$ the algorithm computes $C_j = H(j||m)$ and outputs $\sigma_j = (g_1')^{\frac{1}{y + C_j}}$.

$\mathsf{qSVer}_{PK}(m, j, C, \tau)$ The soft verification algorithm takes in input a message m and an index $j \in \{1, \cdots, q\}$. It computes $C_j = H(j||m_j)$, and checks if $e(\sigma_j, g_2' g_2^{C_j}) = e(g_1', g_2)$. If this is the case, it outputs 1.

$\mathsf{qFake}_{PK,TK}()$ The fake commitment algorithm is the same as qSCom.

$\mathsf{qHEquiv}_{PK,TK}(m_1, \cdots, m_q, j, \mathsf{aux})$ The non-adaptive hard equivocation algorithm uses the trapdoor key TK to hard open a fake commitment (which is originally a commitment to nothing). It computes $C_i = H(i||m_i), \forall i = 1, \cdots, q$ and constructs the polynomial

$$f(z) = \prod_{i=1}^{q}(z + C_i) = \sum_{i=0}^{q} \beta_i z^i.$$

It sets $\alpha = \frac{y}{x}, w = \frac{\alpha'}{f(y)}$ and outputs $\pi = \{\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q\}$.

$\mathsf{qSEquiv}_{PK,TK}(m, j, \mathsf{aux})$ The soft equivocation algorithm is the same as qSOpen.

4.1 Properties of the Scheme

First notice that our commitment scheme is "proper" in the sense of [4]. Recall that a mercurial commitment scheme is said to be "proper" if the soft decommitment is a proper subset of the hard decommitment. In our scheme, a soft decommitment is implicitly contained in a hard one. Indeed, given a hard opening $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$ to a message m at position j and the public key PK, we are able to compute a valid soft decommitment σ_j to the message m of index j.

The correctness of the scheme can be easily verified by inspection. With the next theorem we show that the remaining properties of qTMC are realized as well.

Theorem 1. *Assuming that the Strong Diffie-Hellmann holds for $\mathcal{G}$ and $\mathcal{H}$ is a family of collision resistant hash functions, $\mathcal{QC}$ is a trapdoor q-mercurial commitment scheme.*

Proof (Theorem 1). To prove the theorem we need to make sure that the proposed scheme is binding and hiding, in the sense discussed in section 2.1. We prove each property separately.

q-mercurial binding. To prove the property we need to make sure that neither hard collisions nor soft ones are possible. We prove that it is infeasible to find any of such collisions under the Strong Diffie Hellmann assumption (SDH) for the bilinear group generator $\mathcal{G}$ [3] and the collision resistance of the hash function H.

Let us first consider soft collisions. Next we describe how to adapt the same proof for the case of hard collisions.

Soft collisions. Assume there exists an adversary $\mathcal{A}^{\mathbb{S}}$ that with non-negligible probability ϵ can find a soft collision. We show how to build a simulator $\mathcal{B}^{\mathbb{S}}$ that uses $\mathcal{A}^{\mathbb{S}}$ to solve the q-SDH problem, or to break the collision resistance of H, with probability at least $\epsilon/2$.

$\mathcal{B}^{\mathbb{S}}$ receives in input from its challenger a $(q+3)$-tuple $(g_1, g_1^x, \cdots, g_1^{x^q}, g_2, g_2^x)$ and the description of a hash function H. The simulator runs $\mathcal{A}^{\mathbb{S}}$ on input such values as the public key of the q-mercurial commitment scheme. Then with probability ϵ the adversary outputs (C, m, j, π, m', τ) such that: $C = (g_1', g_2')$ is a commitment, $m \neq m'$, $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$ is a valid hard opening for C to the message m at position j and $\tau = (\sigma_j)$ is a valid soft opening for C to m' of index j. We distinguish two cases:

1. $m \neq m'$ and $C_j = H(j||m) = H(j||m') = C_j'$;
2. $m \neq m'$ and $C_j \neq C_j'$.

At least one of these cases occurs with probability at least $\epsilon/2$. In the first case the simulator immediately has a collision for H. In case 2 we show how to solve the q-SDH problem.

Since $\mathsf{qSVer}_{PK}(m', j, C, \tau) = 1$ we have that $e(\sigma_j, g_2' g_2^{C_j'}) = e(g_1', g_2)$. Moreover, the correct verification of π implies that $g_2' = h^\alpha = g_2^\gamma$ thus $\sigma_j = (g_1')^{\frac{1}{\gamma + C_j'}}$.

Using long division we can write the q-degree polynomial f as $f(z) = \eta(z)(z + C_j') + \eta_{-1}$ where $\eta(z) = \sum_{i=0}^{q-1} \eta_i z^i$ is a polynomial of degree $q-1$ and $\eta_{-1} \in \mathbb{Z}_p$. Thus we can write $\sigma_j = (g_1^{\eta(\gamma)} g_1^{\frac{\eta_{-1}}{\gamma + C_j'}})^w$. Hence first $\mathcal{B}^{\mathbb{S}}$ computes:

$$\delta = (\sigma_j^{1/w} \cdot \prod_{i=0}^{q-1} A_i^{-\eta_i \alpha^i})^{1/\eta_{-1}} = (g_1^{\eta(\gamma)} g_1^{\frac{\eta_{-1}}{\gamma + C_j'}} g_1^{-\eta(\gamma)})^{1/\eta_{-1}} = g_1^{\frac{1}{\gamma + C_j'}}.$$

Finally it computes $\delta^* = \delta^\alpha = g_1^{\frac{\alpha}{\alpha x + C_j'}} = g_1^{\frac{1}{x + C_j'/\alpha}}$ and $C^* = C_j'/\alpha$. The simulator gives (δ^*, C^*) to its challenger. It is easy to see that such pair breaks the q-SDH assumption. Thus with non-negligible advantage $\epsilon/2$ $\mathcal{B}^{\mathbb{S}}$ can break either the q-SDH assumption or the collision resistance of H.

Hard collisions. Let us now assume there exists an adversary $\mathcal{A}^{\mathbb{H}}$ that, given the public key of a q-mercurial commitment scheme, can find a hard collision with non-negligible probability ϵ. Then we construct a simulator $\mathcal{B}^{\mathbb{H}}$ that either solves the q-SDH problem or breaks the collision resistance of H with probability at least $\epsilon/2$. The simulator $\mathcal{B}^{\mathbb{H}}$ is similar to the one described above. The difference is that $\mathcal{A}^{\mathbb{H}}$ outputs: (C, m, j, π, m', π') such that: $C = (g_1', g_2')$ is a commitment, $m \neq m'$ are two different messages, $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$ is a valid hard opening for C to m of index j and $\pi' = (\alpha', w', m_1', \cdots, m_{j-1}', m_{j+1}', \cdots, m_q')$ is a valid hard opening for C to m' of index j. Again we consider two cases:

1. $m \neq m'$ and $C_j = H(j||m) = H(j||m') = C_j'$,
2. $m \neq m'$ and $C_j \neq C_j'$.

Case 1 is the same as before. In case 2, $\mathcal{B}^{\mathbb{H}}$ solves the q-SDH problem as follows. Since $\mathsf{qHVer}_{PK}(m, j, C, \pi) = 1$ and $\mathsf{qHVer}_{PK}(m', j, C, \pi') = 1$, it must be the case that $\alpha = \alpha'$ ($\alpha \neq \alpha'$, would lead to two different g_2' h^α and $h^{\alpha'}$). Moreover, since the commitment scheme is proper from the valid hard opening $\pi' = (\alpha', w', m_1', \cdots, m_{j-1}', m_{j+1}', \cdots, m_q')$ for m_j' we can "extract" a valid soft opening for m_j'. Thus, using exactly the same argument described above, we break the SDH assumption.

Hiding and Equivocation. First notice that, since our scheme is proper, it suffices to check only q-HHEquivocation and q-SSEquivocation hold. In both cases we show that it is infeasible for an adversary to distinguish between a real commitment/decommitment tuple from a fake/equivocation one.

In the q-HHEquivocation game the adversary is asked to tell apart

$$\{(g_1^{f(\gamma)w}, g_2^{\alpha x}), (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)\}$$

from

$$\{(g_1^{\alpha'}, g_2^{y}), (\alpha = \frac{y}{x}, w = \frac{\alpha'}{f(\gamma)}, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)\}$$

In both cases α, w are uniformly random in $\mathbb{Z}_p^*$. This is because, in the first tuple, they are chosen uniformly and at random, while in the second tuple they are distributed, respectively, as y and α', which were chosen uniformly and at random in $\mathbb{Z}_p^*$.

Thus the two distributions are indistinguishable.

The proof of indistinguishability for the q-SSEquivocation is trivial. Indeed, it is easy to see that the elements in the two distributions

$$\{(g_1^{\alpha'}, g_2^{y}), \sigma_i = (g_1')^{\frac{1}{\gamma + C_i}}\}$$

$$\{(g_1^{\alpha'}, g_2^{y}), \sigma_i = (g_1')^{\frac{1}{\gamma + C_i}}\}$$

are distributed in exactly the same manner.

5 Efficiency Considerations

In the previous section we proposed a trapdoor q-mercurial commitment scheme $\mathcal{QC}$ based on the Strong Diffie-Hellmann assumption. In order to build efficient zero knowledge EDB, we also use a trapdoor mercurial commitment scheme $\mathcal{C}$ based on the Discrete Logarithm constructions given in [17,5]. For our convenience we consider an implementation of the scheme that allows us to use some of the parameter already in use for the qTMC scheme. In particular, we use $g_1, A_1 \in G_1$ from the public key of $\mathcal{QC}$ as the public key for $\mathcal{C}$.

Combining the two schemes as described in section 3, we obtain an implementation of zero-knowledge EDB (based on the SDH problem) that allows for proofs that are significantly shorter than those produced by previous proposals.

Below we compare our proposal with the most efficient (in terms of space) implementation known so far, namely the one by Micali *et al.* [17] (MRK from now on, for short), when implemented over elliptic curves with short representation.

We measure efficiency in terms of the space taken by each proof. For both schemes, we assume that the universe $\mathcal{U}$ has size $|\mathcal{U}| = 2^k = q^h$ and, that $q = 2^{k'}$, for simplicity.

Groups Used in the Comparisons. Following [10] we fix a security parameter $\ell = 256$ to achieve $k = 128$ bits of security. Specifically G_1 is realized as a subgroup of points on an elliptic curve E over a finite field $\mathbb{F}_p$ of size p, where p is an ℓ bits prime. If e is a parameter called *embedding degree*, G_2 is a subgroup of $E(\mathbb{F}_{p^e})$ and $G_T \subset E(\mathbb{F}_{p^e}^*)$. In particular we consider elliptic curves with embedding degree $e = 12$ and CM discriminant $D = -3$. As suggested in [10], for the case of *Type 3 groups* (see [10] for details), such parameters enable to obtain elements of G_2 that have size twice the size of elements of G_1.

Table 1. Space required by proofs, in our scheme

q	Membership	Non-membership
2	773	516
4	517	260
8	517	174.7
16	645	132
32	926.6	106.4
64	1455.7	89.3
128	2418.7	77.1
256	4165	68

Bandwidth. A proof of membership in our scheme contains $h(q+4)+5$ elements[8]. A proof of non-membership $4h+4$. In MRK's scheme a proof of membership requires $6k+5$ elements, while a proof of non-membership needs $5k+4$ elements. In both cases all the elements have size ℓ, but, for our scheme, we let q vary. For such a choice of parameters we obtain the following results.

The scheme of Micali *et al.* requires 773 elements for proofs of membership and 644 for proofs of non-membership. Results for our scheme are summarized in Table 1.

Notice that our scheme produces proofs of non-membership, that are always much shorter than the corresponding MRK proofs. The space required by our proofs of membership, on the other hand, compares favorably to MRK scheme only until $q \leq 16$, it gets slightly worse for $q = 32$, and much worse for larger values of q. Thus, the choice of $q = 8$ leads to proofs of membership that are (approximately) 33% shorter, and to proofs of non membership that are almost 73% shorter than MRK!

Notice that such a choice of q (i.e. $q = 8$) keeps the scheme practical also in terms of length of the common reference string. Notice also that, according to our present knowledge of the SDH problem, it seems reasonable to consider the same security parameter for our scheme and for the MRK implementation. This is because Cheon [6] attack requires q to be an upper-bound to a factor of either $p-1$ or $p+1$ in order to be effective. If one sets $q = 8$, as suggested in the table above, this would imply that one should increase the key size of at most 2 bits in the worst case. Thus using the same security parameter for both ours and MRK seems to be reasonable for all practical purposes.

6 Conclusions

In this paper we introduced and implemented the notion of trapdoor q mercurial commitments. Our construction can be used to construct zero knowledge sets that allow for proofs that are much shorter than those obtained by previous

[8] We assume each element has size ℓ. This is because, the size of each element in G_2 is twice that of an element in G_1. Thus whenever an element in G_2 is considered, this counts as two elements in G_1.

work. It would be interesting to investigate if it is possible to come up with an even more efficient implementation of the new primitive. In particular, it would be very interesting to construct a qTMC that allows for openings whose length is independent of q.

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security (1993)
2. Blum, M., De Santis, A., Micali, S., Persiano, P.: Non Interactive Zero Knowledge. SIAM Journal on Computing 20(6) (1991)
3. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, Springer, Heidelberg (2004)
4. Catalano, D., Dodis, Y., Visconti, I.: Mercurial Commitments: Minimal Assumptions and Efficient Constructions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, Springer, Heidelberg (2006)
5. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial commitments with applications to zero-knowledge sets. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, Springer, Heidelberg (2005)
6. Hee Cheon, J.: Security Analysis of the Strong Diffie-Hellman Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, Springer, Heidelberg (2006)
7. Cramer, R., Damgård, I.: New Generation of Secure and Practical RSA-based signatures. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 173–185. Springer, Heidelberg (1996)
8. Damgård, I.: Collision free hash functions and public key signature schemes. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 203–216. Springer, Heidelberg (1988)
9. Dwork, C., Naor, M.: An efficient existentially unforgeable signature scheme and its applications. J. of Cryptology 11(3), 187–208 (1998)
10. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for Cryptographers, Cryptology ePrint Archive, Report 2006/165 (2006), http://eprint.iacr.org/
11. Gennaro, R., Micali, S.: Independent Zero-Knowledge Sets. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, Springer, Heidelberg (2006)
12. Goldwasser, S., Ostrovsky, R.: Invariant Signatures and Non Interactive Zero Knowledge proofs are equivalent. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, Springer, Heidelberg (1993)
13. Lim, C.H.: Efficient Multi-Exponentiation and Application to Batch Verification of Digital Signatures (unpublished manuscript) (August 2000)
14. Lim, C.H.: More Flexible Exponentiation with Precomputation. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, Springer, Heidelberg (1994)
15. Liskov, M.: Updatable zero-knowledge databases. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, Springer, Heidelberg (2005)
16. Merkle, R.: A Digital Signature based on a Conventional Encryption Function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)

17. Micali, S., Rabin, M., Kilian, J.K.: Zero-Knowledge Sets. In: In proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science – FOCS 2003 (2003)
18. Micali, S., Rabin, M., Vadhan, S.: Verifiable Random Functions. In: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science – FOCS 1999 (1999)
19. Ostrovsky, R., Rackoff, C., Smith, A.: Efficient consistency proof on a committed database. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, Springer, Heidelberg (2004)
20. Pedersen, T.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
21. Prabhakaran, M., Xue, R.: Statistically Hiding Sets, Cryptology ePrint Archive, Report 2007/349 (2007), http://eprint.iacr.org/

Strongly Multiplicative Ramp Schemes from High Degree Rational Points on Curves

Hao Chen[1], Ronald Cramer[2], Robbert de Haan[3], and Ignacio Cascudo Pueyo[4]

[1] Software Engineering Institute, East China Normal University, Shanghai 20062, China
haochen@sei.ecnu.edu.cn

[2] CWI, Amsterdam & Mathematical Institute, Leiden University, The Netherlands
http://www.cwi.nl/~cramer

[3] CWI, Amsterdam, The Netherlands
http://www.cwi.nl/~haan

[4] Department of Mathematics, University of Oviedo, Spain
icascudo@orion.ciencias.uniovi.es

Abstract. In this work we introduce a novel paradigm for the construction of ramp schemes with strong multiplication that allows the secret to be chosen in an extension field, whereas the shares lie in a base field. When applied to the setting of Shamir's scheme, for example, this leads to a ramp scheme with strong multiplication from which protocols can be constructed for atomic secure multiplication with communication equal to a linear number of field elements in the size of the network.

This is also achieved by the results from Cramer, Damgaard and de Haan from EUROCRYPT 2007. However, our new ramp scheme has an improved privacy bound that is essentially optimal and leads to a significant mathematical simplification of the earlier results on atomic secure multiplication.

As a result, by considering high degree rational points on algebraic curves, this can now be generalized to algebraic geometric ramp schemes with strong multiplication over a constant size field, which in turn leads to low communication atomic secure multiplication where the base field can now be taken constant, as opposed to earlier work.

1 Introduction

Recent constructions of ramp schemes with (strong) multiplication [2,3] play a crucial role in advances in the communication efficiency of secure multi-party computation [2,3,4] and, quite surprisingly, of constant rate zero knowledge

[1] Hao Chen's research has been supported by NSFC grants 90607005 and 60433050.
[2] Ronald Cramer's research has been partially supported by NWO VICI.
[3] Robbert de Haan's research has been partially funded by the Dutch BSIK/BRICKS project PDC1.
[4] Ignacio Cascudo Pueyo's research has been partially supported by MEC of Spain, project MTM 2004-08115-C04-01 and FPU grant AP2005-0836.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 451–470, 2008.

proofs for circuit satisfiability [9]. The constructions of these dedicated ramp schemes rely on the theory of error correcting codes as well as arithmetic geometry, and allow for the field of definition to be fixed, while offering almost optimal corruption tolerance. This is to be contrasted with Shamir's scheme, where the field size is linear in the size of the network.

In this work we introduce a novel paradigm for the construction of ramp schemes with strong multiplication [5] that allows the secret to be chosen in an extension field, whereas the shares lie in a base field. Our paradigm is based on selection of certain suitable rational subcodes of error correcting codes defined over extension fields.

Applied to the setting of Shamir's scheme, for example, this comes down to choosing random polynomials f subject to the constraints that $f(0)$ is equal to the secret s lying in an extension field L, while the shares $f(P)$ lie in a subfield K. In particular, this appears to be a novel way of turning Shamir's scheme into a ramp scheme with strong multiplication. When applied to the setting of Shamir's scheme, for example, this leads to a ramp scheme with strong multiplication from which protocols can be constructed for atomic secure multiplication with communication equal to a linear number of field elements in the size of the network. This is also achieved by the results from Cramer, Damgaard and de Haan from EUROCRYPT 2007. However, our new ramp scheme has an improved privacy bound (an additive factor linear in the degree of the field extension) that is essentially optimal and it leads to a significant mathematical simplification of the earlier results on atomic secret multiplication.

As a result, by considering high degree rational points on algebraic curves, this can now be generalized to algebraic geometric ramp schemes with strong multiplication over a constant size field, which in turn leads to low communication atomic secure multiplication where the base field can now be taken constant, as opposed to earlier work. This introduces a second scheme with strong multiplication over a constant-sized field, where the previous known such scheme due to Chen and Cramer [2] could be used to perform multiple multiplications in parallel at the cost of one.

For both these algebraic geometric schemes we additionally propose new general zero-error multiparty computation protocols secure against a malicious adversary, with corruption tolerance $t = \Omega(n)$, and where each multiplication in the protocol requires communication of $O(n^3)$ base field elements to perform a multiplication involving up to $\Omega(n)$ base field elements. This matches the communication cost of the low-cost protocol for the special case presented in [4], but requires the use of more involved techniques due to the lack of structure in these general schemes.

2 Prior Work

We first formally define the concept of ramp scheme, which is essentially a non-perfect secret sharing scheme. Ramp schemes are useful because they can achieve a high information rate, i.e., the size of the shares can be much smaller than the size of the secret. We then proceed with a brief reiteration of two strongly multiplicative variants of such ramp schemes, which were presented in [7] and [4]. Both of these ramp schemes are ideal and have a high information rate. In particular they involve secret vectors that consist of k field elements while producing shares that consist of a single field element.

2.1 Ramp Schemes

Let $\mathbb{P} = \{p_1, \ldots, p_n\}$ be a set of players and $\mathcal{A}$ and Γ be two subsets of $2^{\mathbb{P}}$ such that $\Gamma \cap \mathcal{A} = \emptyset$. We define a ramp scheme over the field $\mathbb{F}_q$ as follows.

Let a $d \times e$ matrix M over $\mathbb{F}_q$ and a mapping $\phi : \{1, \ldots, d\} \to \{1, \ldots, n\}$ be given. Given a subset A of $\mathbb{P}$ we denote by M_A the set of the rows M_j of M such that $\phi(j) \in A$.

Definition 1. *The matrix M defines a* ramp scheme *if the following two conditions hold:*

1. *For any $A \in \mathcal{A}$, and any k elements $w_1, \ldots, w_k \in \mathbb{F}_q$, there exists a vector $\boldsymbol{v} \in Ker M_A$ such that its first k coordinates are $w_1, \ldots, w_k$.*
2. *For any $B \in \Gamma$, the i^{th} unit vector $\ _i \in \mathbb{F}_q^e$ is in the image of M_B^T for all $i \in \{1, \ldots, k\}$.*

We say that $\mathcal{A}$ and Γ are the adversary structure and access structure of the scheme, respectively.

To share a secret vector $(s_1, \ldots, s_k)$ with the scheme above, a dealer chooses a random vector $\boldsymbol{v} \in \mathbb{F}_q^e$ such that its first k coordinates are $(s_1, \ldots, s_k)$ and sends to player p_j the elements $M_i v$ for which $\phi(i) = j$. Condition 1 implies that any set of players in the adversary structure can get no information about the secrets, while condition 2 ensures that any set of players in the access structure can reconstruct the secret vector using their shares. Note that the definition in [4] specifies a special case of this definition, where the access and adversary structure are defined by two (different) thresholds.

In the following, let $\odot : \mathbb{F}_q^k \times \mathbb{F}_q^k \to \mathbb{F}_q^k$ be a symmetric non-degenerate bilinear map. We define multiplication of secret shared vectors $\boldsymbol{s}, \boldsymbol{t} \in \mathbb{F}_q^k$ to be via this map, which we denote by $\boldsymbol{s} \odot \boldsymbol{t}$.

Definition 2. *A ramp scheme is* multiplicative *if for any $i \in \{1, \ldots, k\}$, there exist $\lambda_1^{(i)}, \ldots, \lambda_d^{(i)} \in \mathbb{F}_q$ such that for any two secret vectors $\boldsymbol{s}$ and $\boldsymbol{t}$ with sets of shares $(a_1, \ldots, a_d)$ and $(b_1, \ldots, b_d)$, we have that $(\boldsymbol{s} \odot \boldsymbol{t})_i = \sum_{j=1}^{d} \lambda_j^{(i)} a_j b_j$.*

Definition 3. *A ramp scheme is* strongly multiplicative *if it is multiplicative on any subset of players for which the complement is in the adversary structure. In other words, given any $A \in \mathcal{A}$, for any $i \in \{1, \ldots, k\}$ and any j so that $\phi(j) \in \bar{A}$ there exists a $\lambda_j^{(i)}$ in $\mathbb{F}_q$ such that for every two secret vectors $\boldsymbol{s}$ and $\boldsymbol{t}$ with sets of shares $(a_1, \ldots, a_d)$ and $(b_1, \ldots, b_d)$, we have that $(\boldsymbol{s} \odot \boldsymbol{t})_i = \sum_{j:\phi(j) \in \bar{A}} \lambda_j^{(i)} a_j b_j$.*

2.2 Parallel Secure Computation

The first ramp scheme we discuss is due to Franklin and Yung [7]. It has the advantage that, at the price of an additive factor k in the corruption tolerance, we can perform multiplication for k elements in parallel at the cost of a single multiplication.

The ramp scheme works as follows. Let t and k be such that $t + k - 1 < n/2$ and assume that the finite field $\mathbb{F}_q$ is such that $|\mathbb{F}_q| \geq n + k$. Let the sets $\{x_1, \dots, x_n\}$ and $\{e_1, \dots, e_k\}$ be two disjoint sets of distinct elements from $\mathbb{F}_q$. Now for a vector $a = (u_1, \dots, u_k)$ of secret elements from $\mathbb{F}_q$, we select a random polynomial $f(X) \in \mathbb{F}_q[X]$ of degree at most $t + k - 1$ such that $f(e_j) = u_j$ for $j = 1, 2, \dots, k$ and define the shares to be $a_j = f(x_j)$ for $j = 1, 2, \dots, n$.

Clearly, $t + k$ shares or more jointly determine f and hence the secret vector a, so the access structure includes all player sets of size at least $t + k$. As to privacy, it is a straightforward consequence of Lagrange-interpolation that t or fewer shares jointly give no information on the secret vector, so the adversary structure includes all player sets of size at most t. We can sum these properties up by calling the resulting scheme a $(t, t + k)$-*ramp scheme*, with secrets of length k.

Assume that we additionally performed this sharing with a polynomial $g(X)$ for a secret vector $b = (v_1, \dots, v_k)$. Since for $j = 1, 2, \dots, k$ it holds that $(fg)(e_j) = u_j v_j$ and furthermore $(fg)(x_i) = f(x_i)g(x_i)$ for $i = 1, 2, \dots, n$, it follows from Lagrange's interpolation theorem that the scheme is multiplicative. Therefore, we can use the generic method described in [4] to bootstrap a protocol for parallel multiplication from this scheme. For additional details, see [7] or [4].

2.3 Extension Field Multiplication

The other relevant ramp scheme can be found in [4]. With this ramp scheme it is possible to perform multiplications in a finite field using only communication and operations over a subfield, reducing the communication cost of every single multiplication by a multiplicative factor. For the technique to be used it is required that the finite field has a sufficiently large extension degree k over a subfield. Furthermore, the corruption tolerance needs to be decreased by an additive factor $2k$.

The scheme works as follows. Let t and k be such that $t + 2k - 2 < n/2$. A finite field $\mathbb{F}_{q^k} = \mathbb{F}_q(\alpha)$ is selected such that $|\mathbb{F}_q| > n$. Let $x_1, \dots, x_n$ be distinct non-zero elements from $\mathbb{F}_q$, let $a = u_0 + u_1\alpha + \ldots + u_{k-1}\alpha^{k-1} \in \mathbb{F}_{q^k}$ be a secret element and define $u(X) = u_0 + u_1 X + \ldots + u_{k-1}X^{k-1} \in \mathbb{F}_q[X]$. Choose a random polynomial $r(X) \in \mathbb{F}_q[X]$ of degree at most $t - 1$ and define $f(X) = u(X) + r(X) \cdot X^{2k-1} \in \mathbb{F}_q[X]$.

Clearly, since f has degree $t + 2k - 2$, it is clear that $t + 2k - 1$ shares or more jointly determine f and hence the secret vector a. As to privacy, let $u'(X) \in \mathbb{F}_q[X]$ of degree at most $k - 1$ be arbitrary and let $r'(X)$ be the polynomial that evaluates to $r(x_i) + (u(x_i) - u'(x_i))/x_i^{2k-1}$ for t points x_i. Then the polynomial $f'(X) = u'(X) + r'(X) \cdot X^{2k-1}$ is consistent with the evaluation of f in these t

points, but the secret corresponds with $u'(X)$ here. So it is a $(t, t+2k-1)$-ramp scheme, with secrets of length k.

Now, when we multiply two such polynomials $f(X) = u(X) + r(X) \cdot X^{2k-1}$ and $g(X) = v(X) + r'(X) \cdot X^{2k-1}$, the product polynomial fg has as its first $2k-1$ coefficients homogeneous sums $s_k = \sum_{i+j=k} u_i v_j$ of coefficients in $u(X)$ and $v(X)$. It is shown in [4] that this suffices for calculating the coefficients of the secret product in $\mathbb{F}_{q^k}$ via linear functions on the local products of the shares. Therefore, this scheme is also multiplicative and can be used to perform the secure multiplication over $\mathbb{F}_{q^k}$ using shares in $\mathbb{F}_q$. For additional details see [4].

Note that in order to share a secret of length k, the scheme introduces a gap between the privacy and reconstruction thresholds of size $2k-1$, whereas the scheme due to Franklin and Yung only requires a gap of size k. In Section 3 we introduce an improved version of this scheme that matches the latter thresholds.

3 An Initial Observation

A closer examination of the scheme in [4] shows that it uses a secret sharing polynomial that has a fixed k-size gap between the lower degree coefficients that relate to the secret and the higher degree coefficients that introduce randomness. In fact, this explains the disparity between the parameters of the schemes described in Sections 2.2 and 2.3.

The observation described in this section allows to remove this disparity and leads to a scheme with tight parameters that is additionally much easier to describe than the scheme from Section 2.3, while it achieves the same effect. Due to its more natural structure, it additionally generalizes over algebraic geometric curves as demonstrated in Section 4. This leads to low communication atomic secure multiplication protocols where the base field can now be taken constant as opposed to linear in the number of players as required by the approach in [4].

The proposed scheme is based on the following theorem, which generalizes Lagrange's interpolation theorem to a setting where the evaluation points are taken from different extension fields of a perfect base field K while the secret sharing polynomial is taken from $K[X]$. The idea is that the evaluation points get assigned different weights, depending on the extension degree of the smallest extension field of K in which they occur.

Theorem 1. *Let K be a perfect field, and let $\overline{K}$ denote an algebraic closure of K. Fix distinct $a_1, \ldots, a_l \in \overline{K}$ such that there is no pair a_i, a_j $(i \neq j)$ where a_j is a Galois-conjugate (over K) of a_i. For $i = 1, \ldots, l$, let n_i denote $[K(a_i) : K]$, the degree of $K(a_i)$ over K as a field extension, and let N denote $\sum_{i=1}^{l} [K(a_i) : K]$. Then, for each $b_1, \ldots, b_l$ with $b_i \in K(a_i)$ $(i = 1, \ldots, l)$, there exists a unique polynomial $f(X) \in K[X]$ such that deg $(f) < N$ and $f(a_i) = b_i$, $i = 1, \ldots, l$.*

Proof. Let $K[X]_{<N}$ denote the polynomials in $K[X]$ of degree smaller than N. Consider the map

$$\phi : K[X]_{<N} \longrightarrow \bigoplus_{i=1}^{l} K(a_i), f \mapsto (f(a_1), \ldots, f(a_l)).$$

We want to show that ϕ is an isomorphism of K-vector spaces. Since the dimensions on both sides are equal, it is sufficient to argue that ϕ is injective. Indeed, suppose g maps to 0. Then, for $i = 1, \ldots, l$, $g(a_i) = 0$. Since $g \in K[X]$, g must be a multiple of the minimal polynomial h of a_i in $K[x]$. The Galois-conjugates of a_i are the roots of h and hence they are roots of g. Because the field is perfect, h is separable, i.e. all the roots of h are different, and the number of these roots is equal to n_i, so the number of conjugates of a_i is n_i. Note that a_i and a_j are not Galois conjugates for any i, j so g has at least $\sum_{i=1}^{l} n_i = N$ zeroes in $\overline{K}$. Thus, viewing g as an element of $\overline{K}[X]$, we conclude that $g \equiv 0$.

The new scheme works as follows. Let t and k be such that $t + k - 1 < n/2$. A finite field $\mathbb{F}_{q^k} = \mathbb{F}_q[\alpha]$ is selected such that $|\mathbb{F}_q| \geq n$. Let $x_1, \ldots, x_n$ be distinct (not necessarily non-zero) elements from $\mathbb{F}_q$ and select $e \in \mathbb{F}_{q^k}$ such that $[\mathbb{F}_q(e) : \mathbb{F}_q] = k$. The secret sharing is now performed as follows. For a secret element $a \in \mathbb{F}_{q^k}$, we choose a random polynomial $f(X) \in \mathbb{F}_q[X]$ of degree at most $t+k-1$ such that $f(e) = a$. The shares are again $f(x_1), f(x_2), \ldots, f(x_n)$.

Theorem 2. *The previous scheme has $(t + k)$-reconstruction and t-privacy.*

Proof. Reconstruction: Given the value of f in $t+k$ points $x_{i_1}, \ldots, x_{i_{t+k}}$, we can apply the previous theorem with $l = t + k$, $a_j = x_{i_j}$ (so $n_j = 1$ and $N = t + k$), to see that these shares determine the polynomial and hence the secret.

Privacy: Given the value of f in t points $x_{i_1}, \ldots, x_{i_t}$ take in the previous theorem $l = t + 1$, $a_j = x_{i_j}$ for $j = 1, \ldots, l - 1$ and $a_l = e$. Then $n_j = 1$ for $j = 1, \ldots, l - 1$ and $n_l = k$, so $N = t + k$. The theorem shows that for every possible choice of the secret $a \in \mathbb{F}_{q^k}$, there exists a unique polynomial of degree less than $t + k$ such that $f(e) = a$ and f evaluates to the known values in $x_{i_1}, \ldots, x_{i_t}$. △

3.1 Multi-party Computation Secure Against an Eavesdropping Adversary

We can now use this scheme to perform secure multi-party computation of elements in $\mathbb{F}_{q^k}$ using communication and operations over the base field $\mathbb{F}_q$. In particular, when $k = O(n)$, this results in a secure multiplication protocol for which $O(n^2)$ field elements in $\mathbb{F}_q$ need to be communicated, while the multiplication is between elements in $\mathbb{F}_{q^k}$. This corresponds with a communication of only $O(n)$ field elements in $\mathbb{F}_{q^k}$.

The secure multiplication works as follows. Assume that $t + k - 1 < n/2$ and that secrets $a \in \mathbb{F}_{q^k}$ and $b \in \mathbb{F}_{q^k}$ have been secret shared, resulting in shares $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$. Due to Theorem 1 applied to the product polynomial fg there exist constants $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{F}_{q^k}$ such that $f(e)g(e) = \sum_{i=1}^{n} \lambda_i f(x_i)g(x_i)$. Writing this out over the basis $\{1, \alpha, \ldots, \alpha^{k-1}\}$ we find coefficients $\lambda_i^{(j)} \in \mathbb{F}_q$ such that $\pi_j(f(e)g(e)) = \sum_{i=1}^{n} \lambda_i^{(j)} f(x_i)g(x_i)$, where π_j is the map that maps an element $\sum_{j=0}^{k-1} w_{j+1}\alpha^j$ to the coefficient w_j. Now every player p_i reshares the element $\sum_{j=0}^{k-1} \lambda_i^{(j+1)} f(x_i)g(x_i)\alpha^j$, and it is easy to see that due to the linearity of the scheme the players can then locally sum up their new shares to obtain a share in $f(e)g(e)$.

4 Algebraic Geometric Ramp Schemes

Algebraic geometric ramp schemes were first proposed in [2] and later in [3], although the latter scheme is not multiplicative. Here we present a new algebraic geometric ramp scheme, which can be seen as a generalization of the scheme in [4], in the sense that it also allows to perform multiplication over a finite field based on operations in a subfield.

Applying some of the algebraic geometric coding techniques of [2] and using the curves introduced by García and Stichtenoth [8], we can for instance obtain families of curves from which we can define strongly multiplicative ramp schemes with corruption tolerance t with $(1/3-\epsilon)n < t < n/3$ for any $\epsilon > 0$ over constant-sized fields. In particular this implies that we can work with fixed-size shares, i.e., schemes where the share size is independent of the number of players, which was impossible to achieve with the scheme from [4].

We next describe the ramp scheme of [2] and introduce our new algebraic geometric ramp scheme where the dealer uses a high degree rational point on the curve to allocate the secret. Furthermore, we provide proofs that demonstrate both schemes are (strongly) multiplicative given a large enough number of participating players and additionally show how to compute the coefficients corresponding with the (strong) multiplication property.

4.1 Preliminaries

A very nice overview of most of the algebraic geometry theory that is required to describe the results in this paper can be found in [2]. Here we briefly reiterate the key ingredients and in addition introduce the notion of differential form.

Let $\mathbb{F}_q$ be a finite field with algebraic closure $\overline{\mathbb{F}}_q$ and let $\mathcal{C}$ be an absolutely irreducible, projective smooth curve defined over $\mathbb{F}_q$ with genus g. The function field $\overline{\mathbb{F}}_q(\mathcal{C})$ contains elements, called rational functions, which can be seen as maps from the curve $\mathcal{C}$ to $\overline{\mathbb{F}}_q$. The non-zero rational functions have the property that they can have at most a finite number of poles and zeroes, where the number of poles equals the number of zeroes when both are counted with the correct multiplicities.

A divisor is a formal sum $D = \sum_{P\in\mathcal{C}} a_P \cdot P$ with $a_P \in \mathbb{Z}$ for which the support $supp(D)$, i.e., the set of points P for which a_P is nonzero, is finite. Given two divisors $D = \sum_{P\in\mathcal{C}} a_P \cdot P$ and $D' = \sum_{P\in\mathcal{C}} a'_P \cdot P$, we say that $D \geq D'$ if $a_P \geq a'_P$ for all the points P on the curve. The degree of a divisor $D = \sum_{P\in\mathcal{C}} a_P \cdot P$ is the sum of its coefficients, i.e., $deg(D) = \sum_{P\in\mathcal{C}} a_P$.

Every rational function $f \in \overline{\mathbb{F}}_q(\mathcal{C})$ defines a divisor $(f) = \sum_{P\in\mathcal{C}} \nu_P(f) \cdot P$, where ν_P can be seen as a function that counts the number of zeroes or poles of f with the correct multiplicity for every point P. Clearly, $deg(f) = 0$ for every $f \in \overline{\mathbb{F}}_q(\mathcal{C})$.

The set $\Omega(\mathcal{C})$ contains all rational differential forms on $\mathcal{C}$.[1] Every differential form $\eta \in \Omega(\mathcal{C})$ defines a divisor (η), where every pair of such differential forms η, $\eta' \in \Omega(\mathcal{C})$ gives rise to linearly equivalent divisors, i.e., $(\eta') = (\eta) + (f)$ for some $f \in \overline{\mathbb{F}}_q(\mathcal{C})$. Any such divisor K defined by a differential form is called a *canonical divisor*. For any canonical divisor K, we have that $deg(K) = 2g - 2$.

The residue maps $Res_P : \Omega(\mathcal{C}) \to \overline{\mathbb{F}}_q$ assign to every differential form $\eta \in \Omega(\mathcal{C})$ an evaluation in the point $P \in \mathcal{C}$, where $Res_P(\eta) = 0$ if η does not have a pole in P and $Res_P(\eta) \neq 0$ if η has a pole in P of multiplicity one. As with divisors based on rational functions, the multiplicity of a zero or pole in η can be read off from the coefficient at P in the formal divisor sum $(\eta) = \sum_{P \in \mathcal{C}} a_P \cdot P$. Furthermore, the Residue Theorem states that for any $\eta \in \Omega(\mathcal{C})$ we have that $\sum_{P \in \mathcal{C}} Res_P(\eta) = 0$.

For any divisor D, the corresponding Riemann-Roch space $L(D)$ is defined by $L(D) = \{f \in \overline{\mathbb{F}}_q(\mathcal{C}) \mid (f) + D \geq 0\}$. This is a vector space over $\overline{\mathbb{F}}_q$ and its dimension is denoted $\ell(D)$. For any canonical divisor K we have $\ell(K) = g$, and for any divisor D with $deg(D) < 0$ we have that $\ell(D) = 0$. More generally, the Riemann-Roch Theorem states that for any divisor D we have that $\ell(D) = \ell(K - D) + deg(D) - g + 1$. This implies in particular that $\ell(D) = deg(D) - g + 1$ when $deg(D) > 2g - 2$.

Similarly, we can for any divisor D define the space $\Omega(D)$ by

$$\Omega(D) = \{\omega \in \Omega(\mathcal{C})\backslash\{0\} \mid (\omega) + D \geq 0\} \cup \{0\}.$$

There exists an isomorphism $L(K + D) \simeq \Omega(D)$ via the map $f \mapsto f\eta$, where $(\eta) = K$, which allows us to apply the Riemann-Roch Theorem to calculate the dimension of $\Omega(D)$.

An $\mathbb{F}_q$-rational point on $\mathcal{C}$ is a point that can be represented using coordinates in $\mathbb{F}_q$. An $\mathbb{F}_q$-rational divisor is a divisor for which the support is invariant under the Galois group $Gal(\overline{\mathbb{F}}_q/\mathbb{F}_q)$. Note that such a divisor can have support outside of the $\mathbb{F}_q$-rational points on $\mathcal{C}$. The Riemann-Roch space of an $\mathbb{F}_q$-rational divisor admits a basis defined over $\mathbb{F}_q$, and we can consider the $\mathbb{F}_q$-linear span of this basis. We refer to functions in such an $\mathbb{F}_q$-linear span as $\mathbb{F}_q$-rational functions. Similarly, we can define the subset of $\mathbb{F}_q$-rational differential forms in a set $\Omega(\mathcal{C})$. In the sequel all rational functions and differential forms are $\mathbb{F}_q$-rational, unless otherwise specified.

4.2 Interpolation in Riemann-Roch spaces

The following result is the algebraic geometry counterpart of Theorem 1 corresponding with an arbitrary algebraic curve C.

Theorem 3. *Let $P_1, \ldots, P_l$ be points on the curve C such that P_i and P_j are not conjugate for any $i \neq j$. For $i = 1, \ldots, l$ let n_i be the smallest number such*

[1] Rather than formally defining differential forms here, we restrict the description to an overview of their relevant properties. For a formal description of differential forms, the interested reader is referred to [11].

that P_i is $\mathbb{F}_{q^{n_i}}$-rational and define $N = \sum_{i=1}^{l} n_i$. Let G be a rational divisor such that supp $G \bigcap \{P_1, \dots, P_l\} = \emptyset$. Then:

1. *If $N \geq deg(G) + 1$, for any $(y_1, \dots, y_l)$ with $y_i \in \mathbb{F}_{q^{n_i}}$ there exists* at most *one $f \in L(G)$ such that $f(P_i) = y_i$ for all $i = 1, \dots, l$*
2. *If $N \leq deg(G) - 2g + 1$, for any $(y_1, \dots, y_l)$ with $y_i \in \mathbb{F}_{q^{n_i}}$ there exists* at least *one $f \in L(G)$ such that $f(P_i) = y_i$ for all $i = 1, \dots, l$. Furthermore, the number of such rational functions is the same for any $(y_1, \dots, y_l)$.*

Proof. Let $\phi : L(G) \to \bigoplus_{i=1}^{l} \mathbb{F}_{q^{n_i}}$, defined by $f \mapsto (f(P_1), \dots, f(P_l))$. For $i = 1, \dots, l$, let $P_i^{(0)} = P_i, \dots, P_i^{(n_i-1)}$ be the n_i conjugates of P_i under the Frobenius automorphism over $\mathbb{F}_q$. Observe that $\sum_{j=0}^{n_i-1} P_i^{(j)}$ is a rational point, as any element of the group $Gal(\overline{\mathbb{F}}_q/\mathbb{F}_q)$ permutes the conjugates of P_i. Call $A = G - \sum_{i=1}^{n}(\sum_{j=0}^{n_i-1} P_i^{(j)})$. Then $Ker(\phi) = L(A)$. Observe that $deg(A) = deg(G) - N$. Then

1. If $N \geq deg(G) + 1$, $deg(A) < 0$ and $\ell(A) = 0$. Hence ϕ is injective, which proves the property.
2. If $N \leq deg(G) - 2g + 1$, then $deg(A) \geq 2g - 1$ and we can invoke Riemann-Roch theorem to conclude that $l(A) = deg(A) - g + 1 = deg(G) - N - g + 1 = l(G) - N$. We know that $dim(Im\phi) = dim(L(G)) - dim(Ker\phi) = l(G) - l(A) = N$. Therefore ϕ is surjective. △

4.3 An Algebraic Geometric Ramp Scheme with Parallel Multiplication [2]

Let $D = \{Q_1, \dots, Q_k, P_1, \dots, P_n\}$ be a set of $\mathbb{F}_q$-rational points on the curve $\mathcal{C}$ and G be an $\mathbb{F}_q$-rational divisor of degree $2g + t + k - 1$ with support disjoint from D. Note that since G can have support outside the $\mathbb{F}_q$-rational points, it is possible to include all $\mathbb{F}_q$-rational points on $\mathcal{C}$ in D. Every point P_i corresponds to a player p_i and every point Q_j corresponds to the j^{th} position of a secret vector, as follows. To share the secret vector $(s_1, \dots, s_k) \in \mathbb{F}_q^k$ the dealer takes a random rational function $f \in L(G)$ such that $f(Q_i) = s_i$ for all $i = 1, \dots, k$ and sends player p_i the value $f(P_i) \in \mathbb{F}_q$ as his share.

The scheme described above fits into the formal matricial definition of ramp scheme given in Section 2.1, which is useful for the following sections. Let $\{f_1, \dots, f_u\}$ be a basis of $L(G)$ such that $f_i(Q_j) = 1$ if $i = j$ and $f_i(Q_j) = 0$ if $i \neq j$, for $i = \{1, \dots, u\}$ and $j = \{1, \dots, k\}$. It is easy to see that we can always choose such a basis due to Theorem 3. Indeed, we have that $k < deg(G) - 2g + 1 = t + k + 1$ so the theorem ensures the existence of such f_i for $i = 1, \dots, k$. Now simply take $\{f_{k+1}, \dots, f_u\}$ as a basis of $L(G - \sum_{i=1}^{k} Q_i)$, which has dimension $u - k$ according to the Riemann-Roch Theorem.

Next, define the matrix M whose (i, j) entry is $f_j(P_i)$. If we take a vector $\boldsymbol{v} = (s_1, \dots, s_k, r_{k+1}, \dots, r_n)$ and multiply any row of M_i by $\boldsymbol{v}$, we obtain the value $g(P_i)$, where $g = \sum_{j=1}^{k} s_j f_j + \sum_{j=k+1}^{n} r_j f_j$. It holds that $g(Q_i) = s_i$ for any $i = 1, \dots, k$.

Theorem 4. *The description above defines a ramp scheme with t-privacy and $(2g+t+k)$-reconstruction.*

Proof. It can be easily seen as a special case of Theorem 3 that any rational function in $L(G)$ is uniquely determined by its evaluations in $deg(G)+1$ rational points (this is exactly Lemma 1 of [2]). In our case, $deg(G) = 2g+t+k-1$ so any $2g+t+k$ players can reconstruct the rational function and thus the secret vector.

Next we prove privacy. Let A be any set of t players. We only need to argue that for any secret vector $\boldsymbol{s} = (s_1, \ldots, s_k) \in \mathbb{F}_q^k$ there exists a rational function f such that $f(Q_i) = s_i$ and the evaluation of f in the points corresponding to the players in A is zero. Theorem 3 shows us that this is true because $t+k = deg(G) - 2g + 1$. △

4.4 A New Algebraic Geometric Ramp Scheme with Extension Field Multiplication

Let $D = \{P_1, \ldots, P_n\}$ again be a set of $\mathbb{F}_q$-rational points on the curve $\mathcal{C}$ such that $supp(G) \bigcap D = \emptyset$, and additionally let Q be a point on the curve outside the support of G that is $\mathbb{F}_{q^k}$-rational and not $\mathbb{F}_{q^d}$-rational for any integer $d < k$.

Let $\{e_1, e_2, \ldots, e_k\}$ be a basis of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. To share the secret vector $(s_1, \ldots, s_k)$, the dealer selects a random rational function $f \in L(G)$ such that $f(Q) = s_1 e_1 + \ldots + s_k e_k \in \mathbb{F}_{q^k}$, and sends player p_i the value $f(P_i) \in \mathbb{F}_q$ as his share.

We can also represent this ramp scheme by a matrix. In this case we take a basis $\{f_1, \ldots, f_u\}$ of $L(G)$ such that $f_i(Q) = e_i$ for $i = 1, \ldots, k$ and $f_i(Q) = 0$ for $i = k+1, \ldots, n$. It can again be shown that such a basis exists using Theorem 3. We have only one point of degree k and $k \le deg(G) - 2g + 1$, so we know such f_i exist for $i = 1, \ldots, k$, and we can take $\{f_{k+1}, \ldots, f_u\}$ a basis of $L(G - Q - \sum_{i=1}^{k-1} Q_i)$, where $Q_1, Q_2, \ldots, Q_{k-1}$ are the conjugate points of Q under the Frobenius automorphism over $\mathbb{F}_q$.

Let M be the matrix M whose (i,j) entry is $f_j(P_i)$. As before, if we take a vector $\boldsymbol{v} = (s_1, \ldots, s_k, r_{k+1}, \ldots, r_n)$ and multiply any row of M_i by $\boldsymbol{v}$, we obtain the value $g(P_i)$, where $g = \sum_{j=1}^{k} s_j f_j + \sum_{j=k+1}^{n} r_j f_j$ and it holds that $g(Q) = \sum_{i=1}^{k} s_i e_i$.

Theorem 5. *The description above defines a ramp scheme with t-privacy and $(2g+t+k)$-reconstruction.*

Proof. As before, both properties are a direct consequence of Theorem 3. △

4.5 Multiplication

Both of the schemes thus described introduce their own form of multiplication. For the parallel multiplication scheme, given two vectors $\boldsymbol{s} = (s_1, s_2, \ldots, s_k)$ and $\boldsymbol{t} = (t_1, t_2, \ldots, t_k)$, we can define the product $\boldsymbol{s} \odot \boldsymbol{t} = (s_1 t_1, s_2 t_2, \ldots, s_k t_k)$.

For the extension field multiplication scheme, given any two vectors $\boldsymbol{s} = (s_1, s_2, \ldots, s_k)$ and $\boldsymbol{t} = (t_1, t_2, \ldots, t_k)$, representing the elements $s = s_1e_1+s_2e_2+\ldots+s_ke_k \in \mathbb{F}_{q^k}$ and $t = t_1e_1+t_2e_2+\ldots+t_ke_k \in \mathbb{F}_{q^k}$, the product of these two elements in the field $\mathbb{F}_{q^k}$ is some element $u = u_1e_1+u_2e_2+\ldots+u_ke_k \in \mathbb{F}_{q^k}$ for some $u_i \in \mathbb{F}_q$. We can therefore define the product of $\boldsymbol{s}$ and $\boldsymbol{t}$ as $\boldsymbol{s}\odot\boldsymbol{t} = (u_1, u_2, \ldots, u_k)$.

We next prove that, given enough players, the two schemes are multiplicative and strongly multiplicative with regard to their respective multiplications.

THEOREM 6. *The parallel multiplication scheme is multiplicative when* $n \geq 2t + 4g + 2k - 1$ *and strongly multiplicative when* $n \geq 3t + 4g + 2k - 1$.

PROOF. We need to show that for any $i = 1, \ldots, k$ there are coefficients $\lambda_1^{(i)}, \ldots, \lambda_n^{(i)}$ such that for any $f, g \in L(G)$, $f(Q_i)g(Q_i) = \sum_{j=1}^n \lambda_j^{(i)} f(P_j)g(P_j)$. Note that if f and g are in $L(G)$ their product is in the space $L(2G)$.

According to Theorem 3 we have that if $deg(2G) + 1 \leq n$ the mapping $\phi : L(2G) \to \bigoplus_{j=1}^n \mathbb{F}_q$ defined by $h \mapsto (h(P_1), \ldots, h(P_n))$ is linear and injective, so it has an inverse and it is also linear. Furthermore, the maps $\psi_i : L(2G) \to \mathbb{F}_q$ defined by $h \mapsto h(Q_i)$ are also linear for any $i \in \{1, \ldots, k\}$. So the composition of ϕ^{-1} and any ψ_i is linear. Therefore $fg(Q_i)$ is a linear combination of $f(P_j)g(P_j)$ for any f and g in $L(G)$. Finally observe that the condition $deg(2G) + 1 \leq n$ holds whenever $4g + 2t + 2k - 1 \leq n$. △

Similar to the simpler finite field setting the coefficients $\lambda_j^{(i)}$ can be explicitly determined. We now describe how to obtain these using the Residue Theorem (see [11]).

Determining the coefficients $_j^{(i)}$. A consequence of the Residue Theorem is that for any function φ in $L(2G)$ and any differential ω in $\Omega(Q_i + \sum_{j=1}^n P_j - 2G)$ the relation $0 = \sum_{j=1}^n res_{P_j}(\varphi\omega) + res_{Q_i}(\varphi\omega) = \sum_{j=1}^n \varphi(P_j)res_{P_j}(\omega) + \varphi(Q_i)res_{Q_i}(\omega)$ holds. Therefore, if there exists a nonzero element ω in $\Omega(Q_i + \sum_{j=1}^n P_j - 2G)$, applying the theorem for the rational function fg gives a linear relation between the values $fg(Q_i)$ and $fg(P_j)$ for $j = 1, \ldots, n$ for some coefficients which do not depend on f and g. If we can additionally ensure that the coefficient $res_{Q_i}(\omega)$ is non-zero, then we have a relation of the form $fg(Q_i) = \sum_{j=1}^n -\frac{res_{P_j}(\omega)}{res_{Q_i}(\omega)} fg(P_j)$. Thus, $\lambda_j^{(i)} = -\frac{res_{P_j}(\omega)}{res_{Q_i}(\omega)}$ and we are done.

It is a known fact that we can define an isomorphism of $\mathbb{F}_q$-vector spaces $\phi : L(K + Q_i + \sum_{j=1}^n P_j - 2G) \to \Omega(Q_i + \sum_{j=1}^n P_j - 2G)$ defined by $\phi(h) = h\eta$ where K is a canonical divisor and η is a differential such that $div(\eta) = K$. It suffices to find an element h in $L(K + Q_i + \sum_{j=1}^n P_j - 2G)$ with a first order pole in Q_i. Hence, we have to show that there exists an element in the difference of the spaces $L(K + Q_i + \sum_{j=1}^n P_j - 2G)$ and $L(K + \sum_{j=1}^n P_j - 2G)$. Applying the Riemann-Roch theorem for $n \geq 2t + 4g + 2k - 1$ shows us that the dimensions of these spaces differ and the result follows.

THEOREM 7. *The extension field multiplication scheme is multiplicative when* $n \geq 2t + 4g + 2k - 1$ *and strongly multiplicative when* $n \geq 3t + 4g + 2k - 1$.

Proof. Now we need to show that for any $i = 1, \ldots, k$ there exist coefficients $\lambda_1^{(i)}, \ldots, \lambda_n^{(i)}$ in $\mathbb{F}_q$ such that for any $f, g \in L(G)$, $\pi_i(f(Q)g(Q)) = \sum_{j=1}^{n} \lambda_j^{(i)} f(P_j)g(P_j)$. An argument similar to that in Theorem 6 shows that, for $n \geq 2t + 4g + 2k - 1$, there exist elements $r_j \in \mathbb{F}_{q^k}$ such that $f(Q)g(Q) = \sum_{j=1}^{n} r_j f(P_j)g(P_j)$. Now, note that $r_j = \sum_{i=1}^{k} \lambda_j^{(i)} e_i$, which gives us the desired result. △

5 Multi-party Computation Secure Against an Active Adversary

In the sequel we present the techniques that can be used to construct multi-party computation protocols secure against an active adversary for the algebraic geometric ramp schemes presented earlier. Due to the lack of structure in these schemes compared to the simpler polynomial-based approaches we need to introduce some new techniques here. Most of these techniques revolve around the construction of some specialized variants of VSS, which are then employed to ensure that the players honestly participate in the protocol.

6 A VSS Protocol for the Algebraic Geometric Schemes

When the number of players is sufficiently large, we can perform efficient reconstruction of the secret in the presence of corrupted shares. This is due to the strong relation between our schemes and Goppa error correction codes [11]. In both schemes, the set of possible share vectors forms a Goppa code over $\mathbb{F}_q$ of length n (the number of players) with minimum distance larger than or equal to $n - deg(G)$. We know that a code with minimum distance d allows for reconstruction of a codeword in the presence of t errors, provided that $2t + 1 \leq d$. Furthermore, it is known how to efficiently correct such errors for Goppa codes. We have the following property:

Property 1. *Assume that a honest dealer shares a secret vector with one of the algebraic geometric ramp schemes in the previous section. If* $n \geq 3t + 2g + k$, *honest players can efficiently reconstruct the secret vector even when up to* t *corrupted players provide incorrect shares.*

Note that this bound is weaker than that required for strong multiplicativity for any of the two schemes in Section 4.

We now describe the general procedure used to verifiably secret share a vector with a ramp scheme. Recall that in the usual definition of a verifiable secret sharing (VSS), the VSS ensures that at the end of the sharing either all honest players hold consistent shares in a value s or the dealer is disqualified. Additionally, when the dealer is not disqualified, it is guaranteed that the players can uniquely reconstruct the secret s by pooling their shares in s, even when some of the dishonest players provide an incorrect share. We follow this standard definition of VSS, except that we allow the secret to be a vector.

6.1 Definitions and Notation

We need to introduce some new notation. Given an $\mathbb{F}_q$-vector space V with base $\{v_1, \ldots, v_u\}$, consider the tensor product $V \otimes V$. The elements in the space $V \otimes V$ are formal sums $\sum_{i,j} a_{ij}(v_i \otimes v_j)$ with $a_{ij} \in \mathbb{F}_q$. The symmetric tensor $S^2(V)$ is defined to be the subspace consisting of all the elements in $V \otimes V$ such that $a_{ij} = a_{ji}$ for all $i, j \in \{1, \ldots, u\}$.

We define now the space $S^2(L(G))$. Given an element F in this space, we can evaluate it in any pair (P, Q) of points on the curve, where if $F = \sum_{i,j} a_{ij}(f_i \otimes f_j)$ we have $F(P, Q) = \sum_{i,j} a_{ij}(f_i(P) f_j(Q))$. Now, if P_i is the point corresponding to the player p_i, we define F_i to be the rational function in $L(G)$ such that $F_i(P) = F(P_i, P)$.

For the parallel multiplication scheme from Section 4.3 we define F_0 to be the rational function defined by $F_0(P) = F(Q_1, P)$. Furthermore, for the extension field multiplication scheme from Section 4.4, let F_0 be the rational function defined as follows. Take the function $F_0'(Y) = F(Q, Y) = \sum_{i,j} a_{ij} f_i(Q) f_j(Y)$ in the variable Y that runs over the points on the curve. Note that the coefficients $a_{ij} f_i(Q)$ belong to $\mathbb{F}_{q^k}$. Now we can define the rational function $F_0 = \sum_{i,j} \pi_1(a_{ij} f_i(Q)) f_j$, where the function π_1 is the projection function that has been described in Section 3.1. We now have the following symmetry property, which is easily verified.

Proposition 1. *We have that $F_i(P_j) = F_j(P_i)$ and $F_i(Q_1) = F_0(P_i)$ for the parallel multiplication scheme (respectively, $\pi_1(F_i(Q)) = F_0(P_i)$ for the extension field multiplication scheme) for any $F \in S^2(L(G))$ and $i, j \in \{1, \ldots, n\}$.*

6.2 The VSS Scheme

Conceptually, the rational function F_0 plays the same role in the VSS as the secret sharing polynomial does for Shamir's scheme. We now describe how to perform the VSS for the two algebraic geometric schemes.

First, given a secret vector $(s_1, s_2, \ldots, s_k)$ and a divisor D (for our purposes D is always G or $2G$) we define the set $S_{(s_1,\ldots,s_k)}(D) = \{f \in L(D) : f(Q_l) = s_l \; \forall l = 1, \ldots, k\}$ for the parallel multiplication scheme and $S_{(s_1,\ldots,s_k)}(D) = \{f \in L(D) : f(Q) = s_1 e_1 + s_2 e_2 + \cdots + s_k e_k\}$ for the extension field multiplication scheme. The set $S_{(s_1,\ldots,s_k)}(D)$ forms the sets of rational functions from which F_0 can be drawn when the secret vector is $(s_1, \ldots, s_k)$.

Let us also define $\mathcal{S}_{(s_1,\ldots,s_k)}(D) = \{F \in S^2(L(D)) : F_0 \in S_{(s_1,\ldots,s_k)}(D)\}$ for any of both schemes. If the dealer now wants to VSS a vector $(s_1, s_2, \ldots, s_k)$ he must first select a uniformly random element F in $\mathcal{S}_{(s_1,\ldots,s_k)}(G)$ and then send player p_i the rational function $F_i \in L(G)$ for $i = 1, 2, \ldots, n$. After this the players execute a number of steps to ensure the consistency of the data that they received from the dealer. These steps are very similar to those for the Shamir-based VSS described in [5] and we do not enumerate them here. The value $F_i(Q_1) = F(P_i, Q_1) = F_0(P_i)$ (respectively $\pi_1(F_i(Q)) = F_0(P_i)$) should be seen as player $p_i{}'s$ share in the parallel multiplication scheme (respectively the extension field multiplication scheme).

We next prove that this VSS scheme can always be applied and that it provides privacy in the presence of any adversary controlling up to t players. Unique reconstruction of the secret for the honest players follows from an argument similar to that for the Shamir-based VSS scheme and is omitted here.

For the privacy statement, we first assume without loss of generality that the rational share functions F_i that adversarial players receive are all zero. For any subset $B \subset \{P_1, \ldots, P_n\}$ with $|B| = e \leq t$, we define the sets $W_B(D) = \{f \in L(D) : f(P_j) = 0 \ \forall j \in B\}$ and $\mathcal{W}_B(D) = \{F \in S^2(L(D)) : F_j = 0 \ \forall j \in B\}$ respectively denoting the potential secret sharing functions and rational share functions corresponding to this assumption. Note that in particular when $e = 0$, we have $W_B(D) = L(D)$ and $\mathcal{W}_B(D) = S^2(L(D))$.

For the privacy statement to hold, we mainly need to prove that we have $|\mathcal{S}_{(s_1,\ldots,s_k)}(G) \cap \mathcal{W}_B(G)| = |\mathcal{S}_{(s'_1,\ldots,s'_k)}(G) \cap \mathcal{W}_B(G)|$ for any two secret vectors $(s_1, \ldots, s_k)$ and $(s'_1, \ldots, s'_k)$. To prove that the VSS can always be applied we need to prove that $|\mathcal{S}_{(s_1,\ldots,s_k)}(G)| > 0$ for any secret vector $(s_1, \ldots, s_k)$. Both statements can be deduced from the following theorem:

Theorem 8. *For any adversary set B and any secret vector $(s_1, \ldots, s_k)$, the mapping $\mathcal{S}_{(s_1,\ldots,s_k)}(G) \cap \mathcal{W}_B(G) \to S_{(s_1,\ldots,s_k)}(G) \cap W_B(G)$ given by $F \mapsto F_0$ is surjective.*

Proof. We here give the proof for the parallel multiplication scheme. The proof for the extension field multiplication scheme is very similar and therefore omitted here.

Let f be an element of $S_{(s_1,\ldots,s_k)}(G) \cap W_B(G)$. If $s_1 \neq 0$ then take $F = \frac{1}{s_1}(f \otimes f)$. We have that $F_j = \frac{1}{s_1} f(P_j) f = 0$ for any player $p_j \in B$ because $f \in W_B(G)$, so $F \in \mathcal{W}_B(G)$. Moreover $F_0 = \frac{1}{s_1} f(Q_1) f = f$.

If $s_1 = 0$ then select an $h \in L(G)$ such that $h(Q_1) = 1$ and $h(P_j) = 0$ for all $j \in B$. Such h exists due to the privacy properties of the parallel multiplication scheme described in Section 4.3. Now define $F = f \otimes h + h \otimes f$. We have $F_0 = h(Q_1)f + f(Q_1)h = f$ and $F_j = h(P_j)f + f(P_j)h = 0 \ \forall \ P_j \in B$. This completes the proof. △

If we take $B = \emptyset$, Theorem 8 implies that we can always VSS a secret vector since it was already clear from Sections 4.3 and 4.4 that we can always secret share a secret vector. As for the privacy property, observe that as a consequence of the surjectivity of the mapping, for any set B in the adversary structure and any secret vector $(s_1, \ldots, s_k)$ we know that $\mathcal{S}_{(s_1,\ldots,s_k)}(G) \cap \mathcal{W}_B(G)$ is non-empty. Now, given the secret vectors $(s_1, \ldots, s_k)$ and $(s'_1, \ldots, s'_k)$, take any element F in $\mathcal{S}_{(s'_1-s_1,\ldots,s'_k-s_k)}(G) \cap \mathcal{W}_B(G)$. We have that addition by the function F induces a bijective mapping between the sets $\mathcal{S}_{(s_1,\ldots,s_k)}(G) \cap \mathcal{W}_B(G)$ and $\mathcal{S}_{(s'_1,\ldots,s'_k)}(G) \cap \mathcal{W}_B(G)$.

It can be seen, using Property 1 and the proof for the consistency checks in [5], that the VSS protocol additionally guarantees consistency between the shares of the honest players whenever $n \geq 3t + 2g + k$.

7 Low Complexity MPC for Algebraic Geometric Ramp Schemes

In this section we demonstrate multi-party computation protocols secure against an active adversary based on the algebraic geometric ramp schemes from Section 4, where we assume that n is sufficiently large so that we can perform efficient reconstruction of the secret vectors for rational functions in $L(2G)$ and we are also able to perform VSS over $L(2G)$. Concretely, this is ensured when $n \geq 4t + 4g + 2k - 1$.

The protocols in this section require the communication of $O(n^3)$ field elements while operating on vectors consisting of k elements, which matches that attained for the special case detailed in [4]. However, since we lack the convenient structure that the polynomials provided in [4], we required some new specialized forms of VSS to ensure that the players honestly follow the protocol. Below, we provide the details of the special types of VSS that we require.

7.1 Tailored VSS

It is possible to place some restrictions on the randomly selected element $F \in S^2(L(2G))$ that is used for the VSS in order to ensure to the players that the VSS'ed secret vector is of a special form. Here two types of structural restrictions are relevant for our results; one where some positions in the secret vector are fixed to zero and one where all positions in the secret vector contain the same value. We also look at the combination of these two types, where all-zero vectors replace the secret vectors. This particular variant is used to a create a "one-time-pad" that is used to securely verify the equality of the secret vectors in two secret sharings, and is invoked in a slightly different manner as explained below.

We additionally note the following about the special types of VSS before providing the details in the following sections. In Section 7.2, whenever the secret vector is non-zero, the special types of VSS are used to generate rational functions in $L(G)$. On the other hand, when the special VSS is used to generate a one-time-pad, the resulting rational functions are in $L(2G)$. Note that, since $L(G) \subset L(2G)$, we can use a basis for $L(2G)$ of the form $f_1, f_2, \ldots, f_{u'}$, where the rational functions $f_1, f_2, \ldots, f_u$ form the selected basis for $L(G)$. Note also that $S^2(L(G))$ can be embedded in $S^2(L(2G))$ in the natural way.

Fixing zeros and producing repetition. The restriction is imposed as follows for the case where we introduce zero's in the vector such that the first position of the vector remains non-zero. Let $I \subset \{1, \ldots, k\}$ be a set consisting of positions in the vector that should be zero. Let $\{u_v\}$ be a base of $L(2G)$ of the appropriate form as described in Sections 4.3 and 4.4 and $V_I(2G) \subset L(2G)$ be spanned by $\{u_v\}_{v \notin I}$. Then $V_I(2G)$ consists of all functions of $L(2G)$ which are zero in Q_j for $j \in I$. Analogously define $\mathcal{V}_I(2G) = \{F \in S^2(L(2G)) : F_j(Q_l) = 0 \; \forall j = 0, \ldots, n, \; l \in I\}$, which can be seen as a bivariate version of this set. If we now VSS using elements in $\mathcal{V}_I(2G)$ not only does the secret rational function belong to $V_I(2G)$, but so do all rational functions that are received as shares by the players.

We need a similar kind of restriction for the generation of one-time-pads in $L(2G)$ by a certain player p_i, except that in this case we will require that all the elements of $V_I(2G)$ have their first coordinate equal to zero. Therefore given an $F \in \mathcal{V}_I(2G)$, we have that $F_0 = 0$ and this cannot be used as a one-time-pad due to it's lack of randomness. We propose to use the rational function F_i that player p_i receives as his share instead, where the evaluations $F_j(P_i) = F_i(P_j)$ of the players p_j act as the shares in F_i.[2] It remains to show that we can always VSS a random but restricted rational function F_i in this way, and that this procedure does not leak additional useful information to the adversary. The first property is a consequence of the following theorem.

THEOREM 9. *Let B be a set in the adversary structure. The mapping $\mathcal{V}_I(2G) \cap \mathcal{W}_B(2G) \to V_I(2G) \cap W_B(2G)$ given by $F \mapsto F_i$ is surjective.*

The proof is very similar to that of Theorem 8 and omitted here due to space considerations. As a consequence of this theorem, given any rational function $f \in V_I(2G)$ there exists at least one $F \in \mathcal{V}_I(2G)$ such that $F_i = f$. Moreover, the VSS does not add new information to the adversary about F_i, as shown in the following theorem.

THEOREM 10. *Let B be a set in the adversary structure and F any uniformly randomly selected element of $\mathcal{V}_I(2G)$ under the restrictions given above. Then, the values $(F_j)_{j\in B}$ add no further information about F_i to the information given by $F_i(P_j)$.*

PROOF. It suffices to prove that for every rational function $f \in V_I(2G)$ such that $f(P_j) = 0$ for all $j \in B$, we can find an F in $\mathcal{V}_I(2G)$ such that $F_j = 0$ for all j in B and $F_i = f$. This is again a consequence of Theorem 9. △

The repetitive type of structural restriction is only needed for the parallel multiplication scheme and consist of the following. A player wants to VSS a vector $(s, s, \ldots, s)$ of k equal elements in such a way that the coefficients of the rational share functions F_1, F_2 and F_n at the basis elements $f_1, \ldots, f_k$ are also equal.

Let us define the sets $\mathcal{R}_s(D) = \{F \in S^2(L(D)) : F_0(Q_1) = \cdots = F_0(Q_k) = s$ and $F_j(Q_1) = \cdots = F_j(Q_k)\ \forall j = 0, \ldots, n\}$ and $R_s(D) = \{f \in L(D) : f(Q_1) = \cdots = f(Q_k) = s\}$. Privacy and existence can, similar to before, be deduced from the following theorem.

THEOREM 11. *The mapping $\mathcal{R}_s(G) \cap \mathcal{W}_B(G) \to R_s(G) \cap W_B(G)$ given by $F \mapsto F_0$ is surjective for any $s \in \mathbb{F}_q$.*

We omit the proof, as it is very similar to that of Theorems 8 and 9.

Creating a default sharing for $({}^{(1)}_{i}, {}^{(2)}_{i}, \ldots, {}^{(k)}_{i})$. Consider the vector ${}_{i} = (\lambda_i^{(1)}, \lambda_i^{(2)}, \ldots, \lambda_i^{(k)})$. We here create a default ramp sharing of this

[2] This is not known to be possible in the space $L(G)$ as defined here, but any encompassing space $L(G')$ of larger dimension with $supp(G') \cap D = \emptyset$ suffices. In particular this can be done in the space $L(2G)$.

(public) vector that is used later on. To do so, we take the rational function $\lambda_i = \sum_{j=1}^{k} \lambda_i^{(j)} f_j \in L(G)$, such that the share of player p_j is $\lambda_i(P_j)$. Note that in the parallel multiplication scheme $\lambda_i(Q_\ell) = \lambda_i^{(\ell)}$, while in the extension field multiplication scheme $\lambda_i(Q) = \sum_{l=1}^{k} \lambda_i^{(\ell)} e_\ell$. This sharing is later used to create VSS'ed shares in the vector $(\lambda_i^{(1)} y, \lambda_i^{(2)} y, \ldots, \lambda_i^{(k)} y)$ in the space $L(2G)$ from a VSS of y in the space $L(G)$.

7.2 The MPC Protocols Secure Against an Active Adversary

As usual, addition and multiplication with a constant can be performed locally by the players. Therefore, the main focus is on the initialization, secure multiplication and reconstruction parts of the protocol. During the multiplication part of the protocol, the special types of VSS that were introduced in Section 7.1 are used to force the dishonest players to follow the protocol honestly. Due to this, the protocol can basically be seen as an application of the protocol secure against an eavesdropping adversary enhanced with checking information that ensures that players perform the correct steps. We now present the details of the main protocol parts.

Initialization. The dealer verifiably secret shares $\boldsymbol{s}, \boldsymbol{t} \in \mathbb{F}_q^k$ using uniformly random elements $F \in \mathcal{S}_{\boldsymbol{s}}$, $G \in \mathcal{S}_{\boldsymbol{t}}$, resulting in rational functions $f_i := F_i$ and $g_i := G_i$ for every player p_i and secret sharing functions $f_0 := F_0$ and $g_0 := G_0$. For the parallel multiplication scheme we denote $f_{i0} := f_i(Q_1)$ and $g_{i0} := g_i(Q_1)$ and similarly for the extension field multiplication scheme we denote $f_{i0} := \pi_1(f_i(Q))$ and $g_{i0} := \pi_1(g_i(Q))$ for $i, j = 1, 2, \ldots, n$.

Using this notation it is to be understood that f_{i0} is the actual share of player p_i in the scheme based on F and similarly for the g_{i0} and G. Furthermore, for both schemes we denote $f_{ij} := f_i(P_j)$ and $g_{ij} := g_i(P_j)$ for $i, j = 1, 2, \ldots, n$, where the share f_{ij} can be seen as the share of player p_i in the rational function F_j held by player p_j. We also use this convention of using lower case letters to denote the shares and rational functions for the other VSSes introduced in the protocols below.

Multiplication. The following two protocols describe the main parts of the multiplication protocol for the parallel multiplication scheme. After proving their properties, we then sketch the changes required for the extension field multiplication scheme. The general structure of the multiplication protocol is as follows. First, every player p_i simultaneously:

1. Reshares the product $a_i b_i$ of his shares a_i and b_i in the VSS of the secret vectors that are to be multiplied in a special format depending on the scheme involved.
2. Reshares his contribution ${}_i a_i b_i = (\lambda_i^{(1)} a_i b_i, \lambda_i^{(2)} a_i b_i, \ldots, \lambda_i^{(k)} a_i b_i)$ in the output of the multiplication, where the validity of this resharing is verified using the special resharing created in the previous step.

After this the players can add up their shares in the contributions ${}_{\boldsymbol{i}}a_ib_i$ of the players to obtain shares in the product $\boldsymbol{s} \odot \boldsymbol{t} = \sum_{i=1}^{n} {}_{\boldsymbol{i}}a_ib_i$. Below these subprotocols are listed for the respective secret sharing schemes.

Protocol 1: (Parallel multiplication) Resharing the input of player p_i
Input: Two VSSes with elements $F, G \in S^2(L(G))$.
Output: A VSS with $D \in_R \mathcal{R}_s(G)$ *with* $s = f_{i0}g_{i0}$ *or a disqualification for player* p_i.
Protocol:

1. Player p_i VSSes $D \in_R \mathcal{R}_s(G)$.
2. Player p_i VSSes $S \in_R \mathcal{V}_{\{1\}}(2G)$.
3. The players publicly verify that $s - d_0(Q_1) + s_{i0} = 0$. If not, player p_i is disqualified.

Protocol 2: (Parallel multiplication) Computing contribution player p_i
Input: A VSS with $D \in \mathcal{R}_s(G)$.
Output: A VSS with $H^i \in_R \mathcal{S}_{(\boldsymbol{\lambda_i} s)}(G)$ *or a disqualification for player* p_i.
Protocol:

1. The players locally generate shares λ_j in the default sharing of ${}_{\boldsymbol{i}}$.
2. Player p_i VSSes $H^i \in_R \mathcal{S}_{(\boldsymbol{\lambda_i} s)}(G)$.
3. Player p_i VSSes $T \in_R \mathcal{V}_{\{1,2,\ldots,k\}}(2G)$.
4. The players verify that $[\lambda_0 d_0 - h_0^i + t_i](Q_\ell) = 0$ for $\ell = 1, 2, \ldots, k$. If not, player p_i is disqualified.

We now prove that Protocol 1 is private and correct. Since the privacy and correctness proofs for the other protocols are very similar, these are omitted.

Theorem 12. *At the end of Protocol* 1, *either player* p_i *has been disqualified, or the output is a sharing of the correct form.*

Proof. The main claim to be verified is that $d_0(Q_\ell) = f_i(Q_1)g_i(Q_1)$ for $j = 1, 2, \ldots, k$ if player p_i is not disqualified at the end of the protocol. We have $f_i(Q_1)g_i(Q_1) - d_0(Q_1) + s_i(Q_1) = 0$ iff $f_i(Q_1)g_i(Q_1) = d_0(Q_1)$.

Due to the applications of VSS, every player p_j holds a value $[f_ig_i - d_0 + s_i](P_j)$ in the rational function $[f_ig_i - d_0 + s_i]$. The rational function $[f_ig_i - d_0 + s_i]$ and the evaluations held by the players now define a ramp sharing scheme over $L(2G)$ and from our assumptions on the number of players, we know that the players can efficiently and correctly reconstruct the value $[f_ig_i - d_0 + s_i](Q_1)$ from the pooling of their shares. Due to the special VSS structure used for S, the claim now follows. △

Theorem 13. *If player* p_i *is honest, pooling the shares* $f_{ij}g_{ij} - d_0(P_j) + s_{ij}$ *leaks no additional information on* f_ig_i *or* d_0.

Proof. Due to the privacy properties of the secret sharing scheme we can first assume wlog that the shares $(d_j)_{j \in A}$, $(f_{ij}g_{ij})_{j \in A}$, $(s_{ij})_{j \in A}$ of the adversary in the three sharings are all equal to zero. The adversary knows a priori that $f_i g_i \in L(2G) \cap W_B(2G)$, $s_i \in V_{\{1\}}(2G) \cap W_B(2G)$ and $d_0 \in R_s(G) \cap W_B(G)$ for some s he does not know. He also knows that $f_i g_i - d_0 \in V_{\{1\}}(2G) \cap W_B(2G)$. We must prove that pooling the shares, and therefore learning the rational function $h = f_i g_i - d_0 + s_i$, adds no further information to this knowledge.

To do so we prove that for any $d \in R_s(G) \cap W_B(G)$, and $f, g \in L(G)$ such that $fg \in L(2G) \cap W_B(2G)$ and $fg(Q_1) = i(Q_1)$, there exist $F, G \in S^2(L(G))$, $I \in \mathcal{R}_s(G) \cap W_B(G)$ and $S \in \mathcal{V}_{\{1\}}(2G) \cap \mathcal{W}_B(2G)$, such that $s_{i0} = 0$, $f_i = f$, $g_i = g$, $d_0 = d$ and $f_i g_i - d_0 + s_i = h$. As a particular case of Theorem 8 we can see that there exist $F, G \in S^2(L(G))$ and $D \in \mathcal{R}_s(G) \cap \mathcal{W}_B(G)$ such that $f_i = f$, $f_i = g$, $d_0 = d$. Finally take $z = h - f_i g_i + d_0$ which is a rational function in $V_{\{1\}}(2G) \cap W_B(2G)$. As a consequence of Theorem 9 we can show that there exists $S \in \mathcal{V}_{\{1\}}(2G) \cap \mathcal{W}_B(2G)$ with $s_i = z$ and that completes the proof. △

We now briefly describe the adjustments that need to be made to the protocols above in order to obtain equally efficient secure protocols for the extension field multiplication. The most important modification is that whereas in the previously listed protocols every player p_i VSSes the product s of his local shares using an element in $\mathcal{R}_s(G)$, for the extension field multiplication scheme the VSS needs to use an element in $\mathcal{S}_{(s,0,\ldots,0)}$. The reason for this is that in the second protocol this allows multiplication with the public sharing for $_{\boldsymbol{i}}$ in order to locally create a VSS of $_{\boldsymbol{i}}s$ in $L(2G)$, similar to what is done in Protocol 2. The second change, which is also required due to the differing structures of the two schemes, is that in the second scheme the coefficients of the secret vector are accessed via the projection maps $\pi_1, \pi_2, \ldots, \pi_k$, which requires small adjustments in the final verification steps of the two protocols.

Share construction. Every player p_j locally sums his rational function shares H_j^i, resulting in a rational function share $H_j = \sum_{i=1}^{n} H_j^i$ in the product $\boldsymbol{s} \odot \boldsymbol{t}$.

7.3 Complexity Analysis of the Multiplication Protocol

During the multiplication protocol every player performs a constant number of VSSes, where every VSS requires $O(n^2)$ communication. Therefore, the multiplication part requires $O(n^3)$ communication for k elements.

References

1. Blakley, G.R., Meadows, C.: Security of ramp schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1985. LNCS, vol. 196, pp. 242–268. Springer, Heidelberg (1985)
2. Chen, H., Cramer, R.: Algebraic geometric secret sharing schemes and secure multi-party computation over small fields. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 516–531. Springer, Heidelberg (2006)
3. Chen, H., Cramer, R., Goldwasser, S., de Haan, R., Vaikuntanathan, V.: Secure computation from random error correcting codes. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 291–310. Springer, Heidelberg (2007)

4. Cramer, R., Damgaard, I., de Haan, R.: Atomic secure multi-party multiplication with low communication. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 329–346. Springer, Heidelberg (2007)
5. Cramer, R., Damgaard, I., Maurer, U.: General secure multi-party computation from any linear secret sharing scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
6. Cramer, R., Daza, V., Gracia, I., Jiménez Urroz, J., Leander, G., Martí-Farré, J., Padró, C.: On codes, matroids and secure multi-party computation from linear secret sharing schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 327–343. Springer, Heidelberg (2005)
7. Franklin, M., Yung, M.: Communication complexity of secure computation. In: Proceedings of STOC 1992, pp. 699–710. ACM Press, New York (1992)
8. García, A., Stichtenoth, H.: On the asymptotic behavior of some towers of function fields over finite fields. J. Number Theory 61, 248–273 (1996)
9. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of STOC 2007, pp. 21–30. ACM Press, New York (2007)
10. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
11. Tsfasman, M.A., Vlăduţ, S.G.: Algebraic-Geometric Codes. Kluwer, Dordrecht (1991)

Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors

Ronald Cramer[1,2], Yevgeniy Dodis[3], Serge Fehr[2], Carles Padró[4], and Daniel Wichs[3]

[1] Mathematical Institute, Leiden University, The Netherlands
[2] CWI Amsterdam, The Netherlands
{cramer,fehr}@cwi.nl
[3] New York University
{dodis,wichs}@cs.nyu.edu
[4] Universitat Politècnica de Catalunya, Barcelona, Spain
cpadro@ma4.upc.edu

Abstract. Consider an abstract storage device $\Sigma(\mathcal{G})$ that can hold a single element x from a fixed, publicly known finite group $\mathcal{G}$. Storage is private in the sense that an adversary does not have read access to $\Sigma(\mathcal{G})$ at all. However, $\Sigma(\mathcal{G})$ is non-robust in the sense that the adversary can modify its contents by adding some offset $\Delta \in \mathcal{G}$. Due to the privacy of the storage device, the value Δ can only depend on an adversary's *a priori* knowledge of x. We introduce a new primitive called an *algebraic manipulation detection* (AMD) code, which encodes a source s into a value x stored on $\Sigma(\mathcal{G})$ so that any tampering by an adversary will be detected. We give a nearly optimal construction of AMD codes, which can flexibly accommodate arbitrary choices for the length of the source s and security level. We use this construction in two applications:

– We show how to efficiently convert any linear secret sharing scheme into a *robust secret sharing scheme*, which ensures that no *unqualified subset* of players can modify their shares and cause the reconstruction of some value $s' \neq s$.
– We show how to build nearly optimal *robust fuzzy extractors* for several natural metrics. Robust fuzzy extractors enable one to reliably extract and later recover random keys from noisy and non-uniform secrets, such as biometrics, by relying only on *non-robust public storage*. In the past, such constructions were known only in the random oracle model, or required the entropy rate of the secret to be greater than half. Our construction relies on a randomly chosen common reference string (CRS) available to all parties.

1 Introduction

We consider an abstract storage device $\Sigma(\mathcal{G})$ that can hold a single element x from a fixed, publicly known finite (additive) group $\mathcal{G}$. Storage is private in the sense that an adversary does not have read access to $\Sigma(\mathcal{G})$ at all. However, $\Sigma(\mathcal{G})$ allows tampering in the sense that an adversary may manipulate the stored value x by adding some offset $\Delta \in \mathcal{G}$ of his choice. As a result, $\Sigma(\mathcal{G})$ stores the element $x + \Delta \in \mathcal{G}$. Due to the

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 471–488, 2008.

privacy of the storage device, the value Δ can only depend on an adversary's *a priori* knowledge of x. For instance, one-time-pad encryption can be understood as such a storage device: it hides the message perfectly, but an adversary can add (bitwise-xor) a string to the message without being detected. Of course, by itself, this example is not very interesting, since it requires some *additional private and tamper-proof storage* for the one-time pad key.[1] However, in the two applications discussed below, no other private or tamper-proof storage is available and hence we will need to use $\Sigma(\mathcal{G})$ alone to achieve authenticity.

1.1 Linear Secret Sharing Schemes

In a *linear secret sharing scheme* (e.g. Shamir's secret sharing [24] and many others) a secret s is distributed among n players so that each player gets some algebraic *share* of the secret. Any *qualified* subset of the players can pool their shares together and recover s by means of a linear transformation over the appropriate domain while any *unqualified* subset gets no information about s. Unfortunately, the correctness of the recovery procedure is guaranteed only if all the shares are correct. In particular, if a qualified subset of the players pools their shares for reconstruction, but the honest players among them form an unqualified set, then the dishonest players (possibly just one!) can cause the reconstruction of a modified secret. Moreover, the difference between the correct secret s and the reconstructed secret s' is controlled by the corrupted players, due to the linearity of the scheme. Luckily, this is "all" that the corrupted players can do: (1) by the privacy of the secret sharing scheme, the noise introduced by the corrupted players can only depend on their prior knowledge of the secret and (2) by the linearity of the secret sharing scheme, for any attempted modification of their shares, the corrupted players must "know" the additive difference between s and s'. In essence, a linear secret sharing scheme of s can be viewed as storing s on our abstract device $\Sigma(\mathcal{G})$.

To deal with this problem, we introduce the notion of an *algebraic manipulation detection* (AMD) code. This is a probabilistic encoding of a source s from a given set $\mathcal{S}$ as an element of the group $\mathcal{G}$, with unique decodability. The security of the code ensures that, when the encoding is stored in $\Sigma(\mathcal{G})$, any manipulation of contents by an adversary will be detected except with a small error probability δ. The guarantee holds even if the adversary has full a priori knowledge of the source state s. No secret keys are required since we rely on the privacy of $\Sigma(\mathcal{G})$ instead.

Using an AMD code, we can turn any linear secret sharing scheme into a *robust secret sharing scheme* [26], which ensures that no unqualified subset of players can modify their shares and cause the reconstruction of some value $s' \neq s$. The transformation is very simple: apply the linear secret sharing scheme to the encoding of s rather than s itself.

In terms of parameters, we obtain robust secret sharing schemes which are nearly as efficient as their non-robust counterparts, since the overhead added by encoding a source will be very small. More precisely, to achieve security $2^{-\kappa}$, we build an AMD code where the length of the encoding of a u-bit value s is only $2\kappa + \mathcal{O}(\log(u/\kappa))$ bits

[1] For example, by using a slightly longer secret key containing a key to a one-time MAC in addition to the one-time-pad key, one can trivially add authentication to this application.

longer than the length of s. This construction is close to optimal since we prove a lower bound of 2κ on the amount of overhead that an AMD encoding must add to the size of the source. As a concrete example, in order to robustly secret share a 1 megabyte message with security level $\delta = 2^{-128}$, our best construction adds fewer than 300 bits by encoding the message, whereas previous constructions (described below) add nearly 2 megabytes.

Relation to Prior Work on Secret Sharing. Although AMD codes were never formally defined in previous work, some constructions of AMD codes have appeared, mostly in connection with making secret sharing robust [19,6,20]. Although some of these constructions are essentially optimal, all of them are largely inflexible in that the error probability δ is dictated by the cardinality of the source space $\mathcal{S}$: $\delta \approx 1/|\mathcal{S}|$. In particular, this implies that when the cardinality of $\mathcal{S}$ is large, the known constructions may introduce significantly more overhead than what is needed to achieve a particular security threshold. In contrast, our constructions can accommodate arbitrary choices of security δ and message length u.

For example, Cabello, Padró and Sáez [6] (see also [22,21]) proposed an elegant construction of a robust secret sharing scheme which implicitly relies on the following AMD code. For any finite field $\mathbb{F}$ of order q, the encoding of the secret $s \in \mathbb{F}$ is a triple $(s, x, x \cdot s)$, where $x \in_R \mathbb{F}$. This code achieves security $\delta = 1/q$ and optimal message overhead $2\log(q) = 2\log(1/\delta)$ for this value of δ. However, as already mentioned, it is far from optimal when we only desire a security level $\delta \gg 1/q$, making this construction inflexible for many applications.

In the context of robust secret sharing, the inflexibility issue mentioned above has recently been addressed in a paper by Obana and Araki [18], where a *flexible* robust secret sharing scheme (in fact, an AMD code in our terminology) was proposed and claimed to be "proven" secure. However, in the full version of this paper [8], we give an attack on their construction showing it to be completely *insecure*.

1.2 Fuzzy Extractors

A less obvious example comes from the domain of *fuzzy extractors* [10]. A fuzzy extractor extracts a uniformly random key R from some non-uniform secret w (e.g., biometric data) in such a way that this key can be recovered from any w' sufficiently close to w in some appropriate metric space.[2] To accomplish this task, the fuzzy extractor also computes a public *helper string* P in addition to the extracted key R, and then recovers R using w' and P. In their original paper, Dodis et al. [10] constructed fuzzy extractors for the Hamming and several other metrics. Unfortunately, the original notion of a fuzzy extractor critically depends on the value of P being stored on a tamper-proof (though public) device. As observed by Boyen et al. [5,4], this severely limits the usability of the concept. To address this problem, [5,4] introduced a stronger notion of a *robust fuzzy extractor*, where any tampering of P will be detected by the user, even with an imperfect reading w' of w! Thus, P can be stored on a potentially untrusted server without the fear that a wrong key $\tilde{R} \neq R$ will be extracted.

[2] For now and much of the paper, we will concentrate on the Hamming space over $\{0,1\}^n$, later pointing out how to extend our results to related metrics.

Prior Work and Our Result. All of the prior work on robust fuzzy extractors uses some form of a *message authentication code (MAC)* keyed by w to authenticate the public parameters P. Such codes are difficult to construct since w is not a uniformly random secret, and the authentication information needs to be verifiable using an imperfect reading w' of w.

Nevertheless, Boyen et al. [4] gave a generic transformation which makes a fuzzy extractor robust *in the random oracle model*, without considerably sacrificing any of the parameters. In the plain model, Dodis et al. [11] showed how to achieve robustness if the initial secret w contains an entropy rate of at least one half (i.e. the entropy of the secret is at least half the length of the secret). The work of [12] shows that this requirement is necessary for information theoretic security in the plain model, even if no errors are allowed (i.e., $w = w'$). Moreover, when the secret does meet this entropy rate threshold, robustness in [11] is only achieved at a large cost in the length of the extracted random key, as compared to the optimal non-robust extractors for the same entropy threshold.

In this work we take a difference approach and use a portion of the *extracted randomness* R to authenticate the public parameters P. Of course, using a MAC naively is insecure since the adversary who modifies P to $\tilde{P}$ will cause the extraction of some $\tilde{R} \neq R$ and we cannot guarantee that the adversary is unable to produce an authentication tag for $\tilde{P}$ under the key $\tilde{R}$.

We overcome this difficulty by carefully analyzing the effects of modifying the public helper P on the extracted randomness R. We construct fuzzy extractors with a *special linearity property* so that any modification of P into $\tilde{P}$ can be essentially subsumed by giving the attacker the ability to *control* the difference Δ between the original key R extracted from w, P and the "defective" key $\tilde{R} = R + \Delta$ extracted from $w', \tilde{P}$. Thus, on a very high level, storing the public helper P on a public and unprotected storage can be viewed as implicitly storing the extracted key R on our abstract storage device $\Sigma(\mathcal{G})$.

In this application one does not have the freedom of storing some *encoding* of R on $\Sigma(\mathcal{G})$, so AMD codes are not directly applicable. Instead, we introduce a related notion called a *(one-time) message authentication code with key manipulation security* (KMS-MAC). Abstractly, this authentication code is keyed by a random element of some finite group $\mathcal{G}$, and remains secure even if the key is stored in $\Sigma(\mathcal{G})$ so that an adversary can tamper with it by adding an offset Δ. We show how to construct KMS-MACs using appropriate AMD codes.[3] Using a KMS-MAC, we can turn any fuzzy extractor with the above mentioned special linearity property into a robust fuzzy extractor with essentially the same parameters and no restrictions on the entropy rate of the secret w. However, this is (necessarily) done in the *Common Reference String (CRS)* model, as we explain below.

Common Reference String Model. Unfortunately, the impossibility result of [12] guarantees that fuzzy extractors with the special linearity property cannot be constructed in the plain model since they imply robust fuzzy extractors for secrets with

[3] The idea of a KMS-MAC is implicitly used in [11] with a construction that is indeed quite similar to ours. However, the construction there is more complicated since the key is not guaranteed to be uniformly random.

entropy rate below a half. We overcome this pessimistic state of affairs by building such fuzzy extractors (and hence corresponding robust fuzzy extractors) in the *Common Reference String* (CRS) model. The common reference string can be chosen once when the system is designed and can be hardwired/hardcoded into all hardware/software implementing the system. Moreover, the CRS can be published publicly and we allow the attacker to observe (but not modify) it.[4] Our CRS is a random bitstring - it has no trapdoors and we do not require any ability to "program" it. Since most users do not create their own hardware/software but instead assume that a third party implementation is correct, the assumption that this implementation also contains an honestly generated random string does not significantly increase the amount of trust required from users. We do assume that the probability distribution from which the secret w is chosen is independent of the CRS. This is a very natural assumption for biometrics and many other scenarios. However, it also means that our scheme is not applicable in the setting of exposure resilient cryptography (see [9]) where the attacker can learn some function of the secret after seeing the CRS.

What our result shows, however, is that this seemingly minor addition not only allows us to achieve robustness without additional restrictions on the entropy rate of the secret, but also to *nearly match the extracted key length of non-robust fuzzy extractor constructions* (or the robust fuzzy extractor constructions in the random oracle model [4]).

2 Algebraic Manipulation Detection Codes

Definition 1. *An* (S, G, δ)-algebraic manipulation detection code*, or* (S, G, δ)-AMD code *for short, is a* probabilistic encoding *map* $\mathcal{E} : \mathcal{S} \to \mathcal{G}$ *from a set* $\mathcal{S}$ *of size* S *into an (additive) group* $\mathcal{G}$ *of order* G*, together with a (deterministic)* decoding *function* $D : \mathcal{G} \to \mathcal{S} \cup \{\perp\}$ *such that* $D(\mathcal{E}(s)) = s$ *with probability 1 for any* $s \in \mathcal{S}$*. The security of an AMD code requires that for any* $s \in \mathcal{S}$, $\Delta \in \mathcal{G}$, $\Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}] \leq \delta$.

An AMD code is called systematic *if* $\mathcal{S}$ *is a group, and the encoding is of the form*

$$\mathcal{E} : \mathcal{S} \to \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2,\ s \mapsto (s, x, f(x, s))$$

for some function f *and* $x \in_R \mathcal{G}_1$*. The decoding function of a systematic AMD code is naturally given by* $D(s', x', \sigma') = s'$ *if* $\sigma' = f(x', s')$ *and* $\perp$ *otherwise.*

Intuitively, $\mathcal{E}(s)$ can safely be stored on a private storage device $\Sigma(\mathcal{G})$ so that an adversary who manipulates the stored value by adding an offset Δ, cannot cause it to decode to some $s' \neq s$. It is also possible to define a *weak* AMD code where security only holds for a *random* $s \in \mathcal{S}$ rather than an arbitrary one. We focus of regular (strong) AMD codes and mention some constructions and applications of weak AMD codes in the full version of this work [8].

From a practical perspective, it is typically not sufficient to have one particular code, but rather one would like to have a class of codes at hand such that for every choice u

[4] We remark that assuming tamper-proof storage of the CRS, which can be shared by many users, is very different than assuming tamper-proof storage of a "user-specific" helper string P. Indeed, the former can be hardwired into the system, and the latter can not.

for the bit-length of the source s and for every choice κ of the security level, there exists a code that "fits" these parameters. This motivates the following definition:

Definition 2. *An AMD code* family *is a class of AMD codes such that for any $\kappa, u \in \mathbb{N}$ there exists an (S, G, δ)-AMD code in that class with $S \geq 2^u$ and $\delta \leq 2^{-\kappa}$.*

We point out that in this definition, the group $\mathcal{G}$ can be different for every AMD code in the family and is left unspecified. In our constructions the group $\mathcal{G}$ will often be the additive group of the vector space $\mathbb{F}^d$ for some field $\mathbb{F}$. Specifically, we will often focus on the field $\mathbb{F}_{2^d}$ (as an additive group, this is equivalent to $\mathbb{F}_2^d$) so addition (and subtraction) is just bitwise-xor of d bit long strings.

We would like the construction of an AMD code to be close to optimal in that G should not be much larger than S . We consider the *tag size* ϖ of a (S, G, δ)-AMD code defined as $\varpi = \log(G) - \log(S)$. Intuitively, this denotes the number of bits that the AMD code appends to the source. More generally we define the efficiency of an AMD code family as follows.

Definition 3. *The* effective tag size $\varpi^*(\kappa, u)$ *with respect to $\kappa, u \in \mathbb{N}$ of an AMD code family is defined as $\varpi^*(\kappa, u) = \min\{\log(G)\} - u$ where the minimum is over all (S, G, δ)-AMD codes in that class with $S \geq 2^u$ and $\delta \leq 2^{-\kappa}$.*

In the full version of this work [8], we prove the following lower bound on the effective tag size of an AMD code family.

Theorem 1. *Any AMD code family has an affective tag size lower bounded by $\varpi^*(\kappa, u) \geq 2\kappa - 2^{-u+1} \geq 2\kappa - 1$.*

2.1 Optimal and Flexible Construction

We are now ready to present a construction of AMD codes which is both optimal and flexible. As noted in the introduction, a similar, but more complicated construction appeared in [11], though it was presented as part of a larger construction, and its properties were not stated explicitly as a stand-alone primitive. The two constructions were discovered concurrently and independently from each other.

Let $\mathbb{F}$ be a field of size q and characteristic p, and let d be any integer such that $d+2$ is not divisible by p. Define the function $\mathcal{E} : \mathbb{F}^d \to \mathbb{F}^d \times \mathbb{F} \times \mathbb{F}$ by $E(s) = (s, x, f(x, s))$ where

$$f(x, s) = x^{d+2} + \sum_{i=1}^{d} s_i x^i$$

Theorem 2. *The given construction is a systematic $(q^d, q^{d+2}, (d+1)/q)$-AMD code with tag size $\varpi = 2\log q$.*

Proof. We wish to show that for any $s \in \mathbb{F}$ and $\Delta \in \mathbb{F}^{d+2}$: $\Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}] \leq \delta$. It is enough to show that for any $s' \neq s$ and any $\Delta_x, \Delta_f \in \mathbb{F}$: $\Pr[f(x, s) + \Delta_f = f(x + \Delta_x, s')] \leq \delta$. Hence we consider the event

$$x^{d+2} + \sum_{i=1}^{d} s_i x^i + \Delta_f = (x + \Delta_x)^{d+2} + \sum_{i=1}^{d} s'_i (x + \Delta_x)^i \qquad (1)$$

We rewrite the right hand side of (1) as $x^{d+2} + (d+2)\Delta_x x^{d+1} + \sum_{i=1}^{d} s'_i x^i + \Delta_x \cdot p(x)$, where $p(x)$ is some polynomial of degree at most d in x. Subtracting this term from both sides of equation (1), x^{d+2} cancels out and we get

$$-(d+2)\Delta_x x^{d+1} + \sum_{i=1}^{d}(s_i - s'_i)x^i - \Delta_x \cdot p(x) + \Delta_f = 0 \tag{2}$$

We claim that the left side of equation 2 is a *non-zero* polynomial of degree at most $d+1$. To see this, let us consider two cases:

1. If $\Delta_x \neq 0$, then the leading coefficient is $-(d+2)\Delta_x \neq 0$ (here we use the fact that $d+2$ is not divisible by the characteristic of the field).
2. If $\Delta_x = 0$, then (2) simplifies to $\sum_{i=1}^{d}(s_i - s'_i)x^i + \Delta_f = 0$, which is not identically zero since we assumed that $s \neq s'$.

This shows that (2) has at most $d+1$ solutions x. Let B be the set of such solutions so $|B| \leq d+1$. Then

$$\Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}] = \Pr_{x \leftarrow \mathbb{F}}[x \in B] \leq \frac{d+1}{q}$$

□

Notice, the elements of the range group $\mathcal{G} = \mathbb{F}^d \times \mathbb{F} \times \mathbb{F}$ can be conveniently viewed as elements of $\mathbb{Z}_p^t$, for some t (recall, p is the characteristic of $\mathbb{F}$). Thus, addition in $\mathcal{G}$ simply corresponds to element-wise addition modulo p. When $p = 2$, this simply becomes the XOR operation.

Quantifying the above construction over all fields $\mathbb{F}$ and all values of d (such that $d+2$ is not divisible by p), we get a very flexible AMD family. Indeed, we show that the effective tag size of the family is nearly optimal.

Corollary 1. *The effective tag size of the AMD code family is* $\varpi^*(\kappa, u) \leq 2\kappa + 2\log(\frac{u}{\kappa} + 3) + 2$. *Moreover, this can be achieved with the range group* $\mathcal{G}$ *being the group of bitstrings under the bitwise-xor operation.*[5]

We prove the above corollary in the full version of our work [8].

3 Application to Robust Secret Sharing

A *secret sharing scheme* is given by two probabilistic functions. The function Share maps a secret s from some group $\mathcal{G}$ to a vector $S = (S_1, \ldots, S_n)$ where the *shares* S_i are in some group $\mathcal{G}_i$. The function Recover takes as input a vector of shares $\tilde{S} = (\tilde{S}_1, \ldots, \tilde{S}_n)$ where $\tilde{S}_i \in \mathcal{G}_i \cup \{\perp\}$ and outputs $\tilde{s} \in \mathcal{G} \cup \{\perp\}$. A secret sharing schemes

[5] We can also imagine situations where the "base" field $\mathbb{F}'$ of some characteristic p is given to us, and our freedom is in choosing the extension field $\mathbb{F}$ and the appropriate value of d so that $\mathcal{S}$ can be embedded into $\mathbb{F}^d$. Under such restrictions, the effective tag size becomes roughly $2\kappa + 2\log(u) + O(\log p)$.

is defined over some *monotone access structure* which maps subsets $B \subseteq \{1, \ldots, n\}$ to a status: `qualified`, `unqualified`, $\perp$. The correctness property of such a scheme states that for any $s \in \mathcal{G}$ and any *qualified* set B, the following is true with probability 1. If $S \leftarrow \mathsf{Share}(s)$ and $\tilde{S}$ is defined to be $\tilde{S}_i = S_i$ for each $i \in B$ and $\tilde{S}_i = \perp$ for each $i \notin B$, then $\mathsf{Recover}(\tilde{S}) = s$. Similarly, the privacy of such a scheme states that for any *unqualified* subset A, the shares $\{S_i\}_{i \in A}$ reveal no information about the secret s (this is formalized using standard indistinguishability).

Thus, qualified sets of players can recover the secret from their pooled shares, while unqualified subsets learn no information about the secret. Sets of players which are neither qualified nor unqualified might not be able to recover the secret in full but might gain some partial information about its value.

A *linear* secret sharing scheme has the property that the Recover function is linear: given any $s \in \mathcal{G}$, any $S \in \mathsf{Share}(s)$, and any vector S' (possibly containing some $\perp$ symbols), we have $\mathsf{Recover}(S + S') = s + \mathsf{Recover}(S')$, where vector addition is defined element-wise and addition with $\perp$ is defined by $\perp + x = x + \perp = \perp$ for all x.

Examples of linear secret sharing schemes include Shamir's secret sharing scheme [24] where the access structure is simply a threshold on the number of players, or a scheme for a general access structure in [15].

We consider a setting where an honest dealer uses a secret sharing scheme to share some secret s among n players. Later, an outside entity called the *reconstructor* contacts some qualified subset B of the players, collects their shares and reconstructs the secret. The security of the scheme ensures that, as long as the set $A \subseteq B$ of players corrupted by an adversary is unqualified, the adversary gets no information about the shared secret. However, if the *honest* players $B \backslash A$ also form an unqualified subset, then the adversary can enforce the reconstruction of an incorrect secret by handing in incorrect shares. In fact, if the reconstructor contacts a *minimal* qualified subset of the players, then even a single corrupted player can cause the reconstruction of an incorrect secret. Robust secret sharing schemes (defined in [26,3]) ensure that such attacks can't succeed: as long as the adversary corrupts only an unqualified subset of the players, the reconstructor will never recover a modified version of the secret.

Definition 4. *A secret sharing scheme is δ-robust if for any unbounded adversary $\mathcal{A}$ who corrupts an unqualified set of players $A \subseteq \{1, \ldots, n\}$ and any $s \in \mathcal{G}$, we have the following. Let $S \leftarrow \mathsf{Share}(s)$ and $\tilde{S}$ be a value such that, for each $1 \leq i \leq n$,*

$$\tilde{S}_i = \begin{cases} \mathcal{A}(i, s, \{S_i\}_{i \in A}) & \text{if } i \in A \\ S_i \text{ or } \perp & \text{if } i \notin A \end{cases}$$

Then $\Pr[\mathsf{Recover}(\tilde{S}) \notin \{s, \perp\}] \leq \delta$.

We note that in a (non-robust) linear secret sharing scheme, when the adversary modifies shares by setting $\tilde{S}_i = S_i + \Delta_i$ then, by linearity of the scheme, the adversary also knows the difference $\Delta = \tilde{s} - s$ between the reconstructed secret $\tilde{s}$ and the shared secret s. This implies that we can think of s as being stored in an abstract storage device $\Sigma(\mathcal{G})$, which is private for an adversary who corrupts an unqualified subset of the players, yet is not-robust in that the adversary can specify additive offsets so that $\Sigma(\mathcal{G})$ stores $s + \Delta$. This immediately implies that we can turn any linear secret sharing scheme into an δ-robust secret sharing scheme using AMD codes.

Theorem 3. *Let* (Share, Recover) *denote a linear secret sharing scheme with domain* $\mathcal{G}$ *of order* G*, and let* $(\mathcal{E}, D)$ *be an* (S, G, δ)*-AMD code with range* $\mathcal{G}$*. Then the scheme* $(\mathsf{Share}^*, \mathsf{Recover}^*)$ *given by* $\mathsf{Share}^*(s) = \mathsf{Share}(\mathcal{E}(s))$, $\mathsf{Recover}^*(\tilde{S}) = D(\mathsf{Recover}(\tilde{S}))$ *is an* δ*-robust secret sharing scheme.*

Proof. Let $S = \mathsf{Share}^*(S)$ and let $\tilde{S}$ be a vector meeting the requirements of Def. 4. Let $S' = \tilde{S} - S$. The vector S' contains 0 for honest players, $\perp$ for absent players, and arbitrary values for dishonest players. We have:

$$\Pr[\mathsf{Recover}^*(\tilde{S}) \notin \{s, \perp\}] = \Pr[D(\mathsf{Recover}(S) + \mathsf{Recover}(S')) \notin \{s, \perp\}]$$
$$= \Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}]$$

where the value $\Delta = \mathsf{Recover}(S')$ is determined by the adversarial strategy $\mathcal{A}$. By the privacy of the secret sharing scheme, it is only based on the adversary's a-priori knowledge of the shared secret and is otherwise independent of the value $\mathcal{E}(s)$. The conclusion then follows immediately from the definition of AMD codes. □

For Shamir secret sharing (and similar schemes), where the group $\mathcal{G}$ can be an arbitrary field of size $q \geq n$, we can use the optimal and flexible AMD code construction from Section 2.1. In doing so, each player's share would increase by roughly $2\log(1/\delta) + 2\log u$ bits (where u in the length of the message) as compared to the non-robust case.

ROBUST INFORMATION DISPERSAL. Systematic AMD codes have an additional benefit in that the encoding leaves the original value s intact. This could be beneficial in the scenario where players do not care about the privacy of s, but only about its authenticity. In other words, it is safe to use *information dispersal* on s or, alternatively, s can be stored in some public non-robust storage. Using a systematic AMD code which maps s to $(s, x, f(x, s))$, the players can just secret share the authentication information $(x, f(x, s))$ and use it later to authenticate s.Even when the value s is large, the authentication information $(x, f(x, s))$ remains relatively small. Concretely, to authenticate an u-bit secret s, we only need to secret share roughly $2(\log(1/\delta) + \log u)$ bits.

SECURE AND PRIVATE STORAGE / SECURE MESSAGE TRANSMISSION. In some applications we want to make sure that, as long as the honest players form a *qualified* set and the dishonest players form an *unqualified* set, the correct secret will *always* be reconstructed (we do not allow the option of reconstructing $\perp$). This problem is known under the name (unconditional) *secure information dispersal* [23,16] or non-interactive *secure message transmission* [14,13]. There is a generic, though for large player sets computationally inefficient, construction based on a robust secret sharing [7]: for every qualified subset of the involved players, invoke the robust reconstruction until for one set of shares no foul play is detected and a secret is reconstructed. If the robust secret sharing scheme is $1/2^{\kappa+n}$-secure, then this procedure succeeds in producing the correct secret except with probability at most $1/2^{\kappa}$.

ANONYMOUS MESSAGE TRANSMISSION. In recent work [2], Broadbent and Tapp explicitly used the notion of AMD codes introduced in this paper (and our construction of them) in the setting of unconditionally secure multi-party protocols with a dishonest majority. Specifically, AMD codes allowed them to obtain robustness in their protocol for anonymous message transmission. This protocol, and with it the underlying AMD code, was then used in [1] as a building block to obtain a protocol for anonymous quantum communication.

4 Message Authentication Codes with Key Manipulation Security

As a notion related to AMD codes, we define message authentication codes which remain secure even if the adversary can manipulate the key. More precisely, we assume that (only) the key of the authentication code is stored on an abstract private device $\Sigma(\mathcal{G})$ to which the adversary has algebraic manipulation access, but the message and the *authentication tag* are stored publicly and the adversary can modify them at will. This is in contrast to AMD codes where the entire encoding of the message is stored in $\Sigma(\mathcal{G})$.

Definition 5. *An* (S, G, T, δ)-message authentication code with key manipulation security *(*KMS MAC*) is a function* $\mathsf{MAC} : \mathcal{S} \times \mathcal{G} \to \mathcal{T}$ *which maps a* source message *in a set* $\mathcal{S}$ *of size* S *to a* tag *in the set* $\mathcal{T}$ *of size* T *using a* key *from a group* $\mathcal{G}$ *of order* G*. We require that for any* $s \neq s' \in S$*, any* $\sigma, \sigma' \in \mathcal{T}$ *and any* $\Delta \in \mathcal{G}$

$$\Pr[\mathsf{MAC}(s', K + \Delta) = \sigma' \mid \mathsf{MAC}(s, K) = \sigma] \leq \delta$$

where the probability is taken over a uniformly random key $K \in_R \mathcal{G}$*.*

Intuitively, the adversary get some message/tag pair (s, σ). The adversary wins if he can produce an offset Δ and a message $s' \neq s$ along with a tag σ' such that the pair (s', σ') verifies correctly under the key $K + \Delta$. The above definition guarantees that such an attack succeeds with probability at most δ. In fact, the definition is slightly stronger than required, since we quantify over all possible tags σ of the message s (rather than just looking at a randomly generated one). However, since the above definition is achievable and simpler to state, we will consider this stronger notion only. We can also think of a KMS-MAC as a generalization of a standard message authentication code, which only guarantees security for $\Delta = 0$.

As with AMD codes, we will consider the notion of a KMS-MAC family. For efficiency, we are interested in minimizing the tag size $\log(T)$ and the key size $\log(G)$. The following well known lower bounds on standard message authentication codes (e.g., see [25]) obviously also apply to the stronger notion of a KMS-MAC.

Lemma 1. *For any authentication code with security* $\delta \leq 2^{-\kappa}$*, the key size* $\log(G)$ *must be at least* 2κ*, and the tag size* $\log(T)$ *must be at least* κ*.*

We now give a construction of a KMS-MAC out of any systematic AMD code.

Theorem 4. *Let* $\mathcal{E} : \mathcal{S} \to \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2, s \mapsto (s, x, f(x, s))$ *be a systematic* $(|\mathcal{S}|, |\mathcal{S}||\mathcal{G}_1||\mathcal{G}_2|, \delta)$*-AMD code. Then the function* $\mathsf{MAC} : \mathcal{S} \times (\mathcal{G}_1 \times \mathcal{G}_2) \to \mathcal{G}_2$ *yields a* $(|\mathcal{S}|, |\mathcal{G}_1||\mathcal{G}_2|, |\mathcal{G}_2|, \delta)$*-KMS-MAC:*

$$\mathsf{MAC}(s, (x_1, x_2)) = f(x_1, s) + x_2$$

Proof. Assume $K = (x_1, x_2) \in \mathcal{G}_1 \times \mathcal{G}_2$ is chosen uniformly at random, and consider arbitrary $\Delta = (\Delta_1, \Delta_2) \in \mathcal{G}_1 \times \mathcal{G}_2$, $\sigma, \sigma' \in \mathcal{G}_2$, and $s, s' \in \mathcal{S}$, where $s \neq s'$.

The event $\mathsf{MAC}(s, K) = \sigma$ is the event $f(x_1, s) + x_2 = \sigma$, which is the same as $x_2 = -f(x_1, s) + \sigma$. Let us call this event E_1. Similarly, the event $\mathsf{MAC}(s', K + \Delta) = \sigma'$ is the event $f(x_1 + \Delta_1, s') + (x_2 + \Delta_2) = \sigma'$, which is the same as $f(x_1 + \Delta_1, s') = -x_2 + \sigma' - \Delta_2$. Let us call this event E_2. Thus, we need to bound $\Pr[E_2 \mid E_1]$.

Let us denote $\Delta_f = -\sigma + \sigma' - \Delta_2$ and define an auxiliary event E_2' as $f(x_1 + \Delta_1, s') = f(x_1, s) + \Delta_f$. We claim that $\Pr[E_2 \mid E_1] = \Pr[E_2' \mid E_1]$. Indeed, if $x_2 = -f(x_1, s) + \sigma$, then

$$-x_2+\sigma'-\Delta_2 = -(-f(x_1,s)+\sigma)+\sigma'-\Delta_2 = f(x_1,s)+(-\sigma+\sigma'-\Delta_2) = f(x_1,s)+\Delta_f$$

Finally, notice that E_2' and E_1 are *independent*. Indeed, since E_2' does not depend on x_2, and x_2 is chosen at random from $\mathcal{G}_2$, whether or not x_2 is equal to $-f(x_1, s) + \sigma$ does not affect any other events not involving x_2. Thus, $\Pr[E_2' \mid E_1] = \Pr[E_2']$. Therefore, we have

$$\Pr[\mathsf{MAC}(s', K+\Delta) = \sigma' \mid \mathsf{MAC}(s, K)=\sigma] = \Pr[f(x_1+\Delta_1, s') = f(x_1, s)+\Delta_f] \leq \delta$$

where the last inequality follows directly from the security of the AMD code, since $s \neq s'$. □

Using the systematic AMD code family constructed in Section 2.1, we get a nearly optimal KMS-MAC family. In particular, plugging in the systematic AMD code family from Theorem 2 and using the parameters obtained in Corollary 1, we get:

Corollary 2. *There is a KMS-MAC family such that, for any* $\kappa, u \in \mathbb{N}$*, the family contains an* (S, G, T, δ)*-KMS-MAC (with respect to XOR operation) with* $\delta \leq 2^{-\kappa}$*,* $S \geq 2^u$ *and*

$$\log(G) \leq 2\kappa + 2\log(u/\kappa + 3) + 2$$
$$\log(T) \leq \kappa + \log(u/\kappa + 3) + 1$$

5 Application to Robust Fuzzy Extractors

We start by reviewing the some basic definitions needed to define the notion of fuzzy extractors from [10].

MIN-ENTROPY. The *min-entropy* of a random variable X is $\mathbf{H}_\infty(X) = -\log(\max_x \Pr_X[x])$. Following [10], we define the (average) conditional min-entropy of X given Y as $\widetilde{\mathbf{H}}_\infty(X \mid Y) = -\log(\mathbf{E}_{y \leftarrow Y}(2^{-\mathbf{H}_\infty(X|Y=y)}))$ (here the expectation is taken over y for which $\Pr[Y = y]$ is nonzero). This definition is convenient for cryptographic purposes, because the probability that the adversary will predict X given Y is $2^{-\widetilde{\mathbf{H}}_\infty(X|Y)}$. Finally, we will use [10, Lemma 2.2], which states that $\widetilde{\mathbf{H}}_\infty(X \mid Y) \geq \mathbf{H}_\infty((X, Y)) - \lambda$, where 2^λ is the number of elements in Y.

SECURE SKETCHES. Let $\mathcal{M}$ be a metric space with distance function dis. Informally, a secure sketch enables recovery of a string $w \in \mathcal{M}$ from any "close" string $w' \in \mathcal{M}$ without leaking too much information about w.

Definition 6. *An* (m, m', t)-secure sketch *for a metric space* $\mathcal{M}$ *is a pair of efficient randomized procedures (*$\mathsf{SS}, \mathsf{Rec}$*) s.t.:*

1. *The sketching procedure* SS *on input* $w \in \mathcal{M}$ *returns a bit string* $s \in \{0,1\}^*$*. The recovery procedure* Rec *takes an element* $w' \in \mathcal{M}$ *and* $s \in \{0,1\}^*$*.*
2. Correctness: *If* $\mathsf{dis}(w, w') \leq t$ *then* $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$*.*
3. Security: *For any distribution* W *over* $\mathcal{M}$ *with min-entropy* m*, the (average) min-entropy of* W *conditioned on* s *does not decrease very much. Specifically, if* $\mathbf{H}_\infty(W) \geq m$ *then* $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \geq m'$*.*

The quantity $m - m'$ *is called the* entropy loss *of the secure sketch.*

As already mentioned in Footnote 2, we will concentrate on the Hamming metric over $\{0,1\}^n$, later extending our results to several related metrics. For this metric we will make use of the *syndrome construction* from [10]. For our current purposes, though, we only need to know that this construction is a *linear transformation* over $\mathbb{F}_2^n$.

STATISTICAL DISTANCE. Let X_1, X_2 be two probability distributions over some space S. Their *statistical distance* is $\mathbf{SD}(X_1, X_2) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{s \in S} |\Pr_{X_1}[s] - \Pr_{X_2}[s]|$. If $\mathbf{SD}(X_1, X_2) \leq \varepsilon$, we say they are ε-close, and write $X_1 \approx_\varepsilon X_2$. Note that ε-close distributions cannot be distinguished with advantage better than ε even by a computationally unbounded adversary. We use the notation U_d to denote (fresh) uniform distribution over $\{0,1\}^d$.

RANDOMNESS EXTRACTORS FOR AVG. MIN ENTROPY. A randomness extractor, as defined in [17], extracts a uniformly random string from any secret with high enough entropy using some randomness as a seed. Here we include a slightly altered definition to ensure that we can extract randomness from any secret with high enough *average* min-entropy.

Definition 7. *A function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ *is called a* (m, ℓ, ε)-extractor *if for all random variables* X *and* Y *such that* $X \in \{0,1\}^n$ *and* $\widetilde{\mathbf{H}}_\infty(X \mid Y) \geq m$*, and* $I \leftarrow U_d$*, we have* $\mathbf{SD}(\ (Y, \mathsf{Ext}(X; I), I)\ ,\ (Y, U_\ell, U_d)\ ) \leq \varepsilon$.

It was shown by [10, Lemma 2.4] that universal hash functions are good extractors in the above sense. In particular, the construction $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\ell$, defined by $\mathsf{Ext}(x, i) \stackrel{\text{def}}{=} [x \cdot i]_1^\ell$ is a (m, ℓ, ε)-extractor for any $\ell \leq m - 2\log(1/\varepsilon)$. Here the multiplication $x \cdot i$ is performed in the field $\mathbb{F}_{2^n}$ and the notation $[z]_1^\ell$ denotes the first ℓ bits of z.

FUZZY EXTRACTORS. A fuzzy extractor extracts a uniformly random key from some secret w in such a way that the key can be recovered from any w' close to w. The notion was first defined in [10]. Here we alter the definition to allow for a public common reference string (CRS).

Definition 8. *An* $(m, \ell, t, \varepsilon)$-fuzzy extractor *for a metric space* $\mathcal{M}$ *is defined by randomized procedures* $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ *with the following properties:*

1. *The procedure* Init *takes no inputs and outputs a string* $\mathrm{CRS} \in \{0,1\}^*$.
2. *The generation procedure* Gen*, on input* $w \in \mathcal{M}, \mathrm{CRS} \in \{0,1\}^*$*, outputs an extracted string* $R \in \{0,1\}^\ell$ *and a helper string* $P \in \{0,1\}^*$*. The reproduction procedure* Rep *takes* $w' \in \mathcal{M}$ *and* $P, \mathrm{CRS} \in \{0,1\}^*$ *as inputs. It outputs* $\tilde{w} \in \mathcal{M} \cup \{\perp\}$.
3. Correctness: *If* $\mathsf{dis}(w, w') \leq t$ *and* $(R, P) \leftarrow \mathsf{Gen}(w, \mathrm{CRS})$*, then* $\mathsf{Rep}(w', P, \mathrm{CRS}) = R$.
4. Privacy: *For any distribution* W *with min-entropy* m *over the metric* $\mathcal{M}$ *, the string* R *is close to uniform even conditioned on the value of* P*. Formally, if* $\mathbf{H}_\infty(W) \geq m$ *and* $(R, P) \leftarrow \mathsf{Gen}(W, \mathrm{CRS})$*, then* $(R, P, \mathrm{CRS}) \approx_\varepsilon (U_\ell, P, \mathrm{CRS})$.

Composing an (m, m', t)-secure sketch with a (m', ℓ, ε)-extractor $\mathsf{Ext}\colon \mathcal{M} \times \{0,1\}^d \to \{0,1\}^\ell$ (as defined in Def. 7) yields a $(m, \ell, t, \varepsilon)$-fuzzy extractor [10]. The construction of [10] has an empty CRS and sets $P = (\mathsf{SS}(w), i)$ and $R = \mathsf{Ext}(w; i)$ for a random i. However, it is easy to see that the construction would remain secure if the extractor seed i was contained in the CRS and P was just $\mathsf{SS}(w)$. One advantage of such approach would be that the Gen and Rep algorithms are then deterministic which might make them easier to implement in hardware. Another advantage is that it would eventually allow us to overcome the impossibility barrier of robust fuzzy extractors (defined next) in the plain model.

5.1 Definition of Robust Fuzzy Extractor in CRS Model

Fuzzy extractors allow one to reveal P publicly without sacrificing the security of the extracted randomness R. However, there are no guarantees when an active attacker modifies P. To prevent such attacks, robust fuzzy extractors were defined and constructed in [4,11]. Here we define robust fuzzy extractors in the CRS model.

For two (correlated) random variables W, W' over a metric space $\mathcal{M}$, we say $\mathsf{dis}(W, W') \leq t$ if the distance between W and W' is at most t with probability one. We call (W, W') a (t, m)-*correlated pair* if $\mathsf{dis}(W, W') \leq t$ and $\mathbf{H}_\infty(W) \geq m$. It will turn out that we can get more efficient constructions if we assume that the random variable $\Delta = W - W'$ indicating the errors between W and W' is independent of W (this was the only case considered by [4]). However, we do not want to make this assumption in general since it is often unlikely to hold. We define the family $\mathcal{F}^{all}_{t,m}$ to be the family of all (t, m)-correlated pairs (W, W') and the family $\mathcal{F}^{indep}_{t,m}$ to be the family of (t, m)-correlated pairs for which $\Delta = W - W'$ is independent of W.

Definition 9. *An* $(m, \ell, t, \varepsilon, \delta)$-robust fuzzy extractor *for a metric space* $\mathcal{M}$ *and a family* $\mathcal{F}$ *of* (t, m)*-correlated pairs is an* $(m, \ell, t, \varepsilon)$*-fuzzy extractor over* $\mathcal{M}$ *such that for all* $(W, W') \in \mathcal{F}$ *and all adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{c}\mathsf{Rep}(\tilde{P}, w', \mathrm{CRS}) \neq \perp \\ \tilde{P} \neq P\end{array}\middle|\begin{array}{c}\mathrm{CRS} \leftarrow \mathsf{Init}(),\ \ (w, w') \leftarrow (W, W') \\ (P, R) \leftarrow \mathsf{Gen}(w, \mathrm{CRS}),\ \ \tilde{P} \leftarrow \mathcal{A}(P, R, \mathrm{CRS})\end{array}\right] \leq \delta$$

We call the above notion post-application *robustness and it will serve as our main definition. We also consider a slightly weaker notion, called* pre-application *robustness where we do not give* R *to the adversary* $\mathcal{A}$.

The distinction between *pre*-application and *post*-application robustness was already made in [4,11]. Intuitively, when a user Alice extracts a key using a robust fuzzy extractor, she may use this key for some purpose such that the adversary can (partially) learn the value of the key. The adversary can then mount an attack that modifies P based on this learned value. For post-application security, we insist that robustness is preserved even in this setting. For pre-application security, we assume that the adversary has no partial information about the value of the key.

5.2 Construction

We are now ready to construct robust fuzzy extractors in the CRS model. First, let us outline a general idea for the construction using an extractor Ext, a secure sketch (SS, Rec) and a one-time (information-theoretic) message authentication code MAC. A pictorial representation of the construction is shown in Figure 1 and pseudo-code is given below.

> $\mathsf{Init}()$ outputs a random seed i for the extractor Ext as a shared CRS.
> $\mathsf{Gen}(w, i)$ does the following:
> $\quad R \leftarrow \mathsf{Ext}(w, i)$ which we parse as $R = (R_{mac}, R_{out})$.
> $\quad s \leftarrow \mathsf{SS}(w)$, $\sigma \leftarrow \mathsf{MAC}(s, R_{mac})$, $P := (s, \sigma)$.
> $\quad$ Output (P, R_{out}).
> $\mathsf{Rep}(w', \tilde{P}, i)$ does the following:
> $\quad$ Parse $\tilde{P} = (\tilde{s}, \tilde{\sigma})$. Let $\tilde{w} \leftarrow \mathsf{Rec}(w', \tilde{s})$. If $d(\tilde{w}, w') > t$ then output $\perp$.
> $\quad$ Using $\tilde{w}$ and i, compute $\tilde{R}$ and parse it as $\tilde{R}_{out}$, $\tilde{R}_{mac}$.
> $\quad$ Verify $\tilde{\sigma} = \mathsf{MAC}(\tilde{s}, \tilde{R}_{mac})$. If equation holds output $\tilde{R}_{out}$, otherwise output $\perp$.

The idea is fairly intuitive. First, we extract randomness from w using the public extractor seed i. Then we use part of the extracted randomness R_{out} as the output, and the remaining part R_{mac} as the key for the one-time information-theoretic MAC to authenticate the secure sketch s of w.

However, in arguing robustness of the reconstruction phase, we notice that there is a problem. When an adversary modifies s to some value $\tilde{s}$ then this will force the user to incorrectly recover $\tilde{w} \neq w$, which in turn leads to the reconstruction of $\tilde{R} \neq R$ and $\tilde{R}_{mac} \neq R_{mac}$. So the key $\tilde{R}_{mac}$, which is used to verify the authenticity of s, will itself be modified when s is!

To break the circularity, we will need to use a special linearity property of the fuzzy extractor. Namely, we want yo make sure that an adversary who modifies s to $\tilde{s}$ will know the offset $R^{\tilde{\Delta}}_{mac} = \tilde{R}_{mac} - R_{mac}$. We formalize this as follows.

FUZZY EXTRACTOR LINEARITY PROPERTY: *For any w, w', i, let $\Delta = w' - w$, $s = \mathsf{SS}(w)$, $R = \mathsf{Ext}(w, i)$. For any $\tilde{s}$, let $\tilde{w} = \mathsf{Rec}(w', \tilde{s})$ and $\tilde{R} = \mathsf{Ext}(\tilde{w}, i)$. Then, there is a deterministic function g such that $R^{\tilde{\Delta}} = \tilde{R} - R = g(\Delta, s, \tilde{s}, i)$.*

It is easy to show that, using the syndrome based construction of a secure sketch and the extractor $\mathsf{Ext}(x, i) \stackrel{\text{def}}{=} [x \cdot i]_1^{\ell}$, the resulting fuzzy extractor satisfies the above linearity property. In the full version of this paper, we give a more general treatment

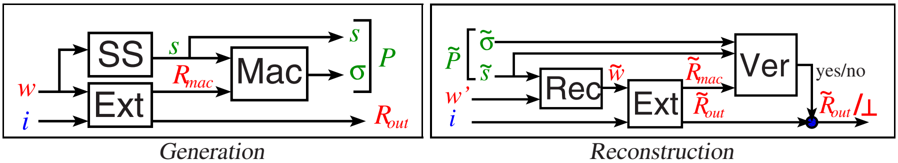


Fig. 1. Construction of Robust Fuzzy Extractor

showing that many other natural secure sketch and extractor constructions produce a fuzzy extractor with the above property.

Given a fuzzy extractor with the above linearity property, we can can think of R_{mac} as being stored in an abstract device $\Sigma(\mathcal{G})$ which is private but only weakly robust in that the adversary can specify an additive offset by modifying s. We can then use a KMS-MAC which remains secure even when the key is stored on such a device. Hence, the adversary will not be able to come up with a valid pair $(\tilde{s}, \tilde{\sigma})$ where $\tilde{s} \neq s$. We formalize this intuition in the next section.

5.3 Security of Construction and Parameters

We are now show that the construction outlined in Section 5.2 indeed satisfies the definition of a robust fuzzy extractor. Let $(\mathsf{SS}, \mathsf{Rec})$ be a (m, m', t)-secure sketch and let u be an upper bound on the size of $\mathsf{SS}(w)$. Let MAC be a (S, G, T, δ)-KMS-MAC, such that $S \geq 2^u$. Assume that the keys for MAC come from a group $\mathcal{G} = \{0,1\}^k$ under the XOR operation so that $G = 2^k$. Let $\mathcal{F}$ be a class of (t, m)-correlated variables (W, W') and let $\hat{m}$ be the largest value such that $\hat{m} \leq \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), W - W')$ for any $(W, W') \in \mathcal{F}$. Let Ext be a $(\hat{m}, \ell, \varepsilon)$-strong randomness extractor seeded by randomness i of length d. Lastly, assume that our secure sketch and randomness extractor produce a fuzzy extractor which satisfies the above defined fuzzy extractor linearity property.

Theorem 5. *When instantiated with the primitives* Ext, MAC *and* $(\mathsf{SS}, \mathsf{Rec})$*, our construction yields a* $(m, \ell - k, t, 2\varepsilon, \delta + \varepsilon)$*-robust-fuzzy extractor for the family* $\mathcal{F}$*.*

Proof. The correctness property of the fuzzy extractor is guaranteed by the correctness of the secure sketch. The privacy property follows from the security of the randomness extractor. Recall, that the adversary can observe i, s, σ. Since, by definition, $\hat{m} \leq \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W))$ the distribution (i, s, R_{mac}, R_{out}) can be distinguished from $(i, s, U_k, U_{\ell-k})$ with probability at most ε. In particular,

$$(i, s, R_{mac}, R_{out}) \approx_\varepsilon (i, s, U_k, U_{\ell-k}) \approx_\varepsilon (i, s, R_{mac}, U_{\ell-k})$$

and so $(i, s, R_{mac}, R_{out}) \approx_{2\varepsilon} (i, s, R_{mac}, U_{\ell-k})$ by the triangle inequality. An adversary given i, s, σ is weaker than an adversary given i, s, R_{mac} and even this latter adversary can distinguish R_{out} from $R_{\ell-k}$ with probability at most 2ε.

For robustness, consider any pair $(W, W') \in \mathcal{F}$ and any adversary $\mathcal{A}$ attacking the robustness of the scheme. Then

$$\Pr[\mathcal{A}\text{ succeeds}] = \Pr\left[\begin{array}{c}\mathsf{Rep}(\tilde{P}, w', \mathrm{CRS}) \neq \bot\\ \text{and } \tilde{P} \neq P\end{array}\middle|\begin{array}{c}\mathrm{CRS} \leftarrow \mathsf{Init}(),\ (w, w') \leftarrow (W, W')\\ (P, R) \leftarrow \mathsf{Gen}(w, \mathrm{CRS})\\ \tilde{P} \leftarrow \mathcal{A}(\mathrm{CRS}, P, R)\end{array}\right]$$

$$= \Pr\left[\begin{array}{c}\mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma}\\ \\ (\tilde{s}, \tilde{\sigma}) \neq (s, \sigma)\end{array}\middle|\begin{array}{c}i \leftarrow U_d,\ (w, w') \leftarrow (W, W')\\ (R_{mac}, R_{out}) := \mathsf{Ext}(w, i)\\ s := \mathsf{SS}(w),\ \sigma := \mathsf{MAC}(s, R_{mac})\\ (\tilde{s}, \tilde{\sigma}) \leftarrow \mathcal{A}(i, s, \sigma, R_{out})\\ \tilde{w} := \mathsf{Rec}(w', \tilde{s}),\ (\tilde{R}_{mac}, \tilde{R}_{out}) := \mathsf{Ext}(\tilde{w}, i)\end{array}\right]$$

Now we use the fuzzy extractor linearity property which defines the deterministic function g such that

$$\Pr[\mathcal{A}\text{ succeeds}] = \Pr\left[\begin{array}{c}\mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma}\\ (\tilde{s}, \tilde{\sigma}) \neq (s, \sigma)\end{array}\middle|\begin{array}{c}i \leftarrow U_d,\ (w, w') \leftarrow (W, W')\\ (R_{mac}, R_{out}) := \mathsf{Ext}(w, i)\\ s := \mathsf{SS}(w),\ \sigma := \mathsf{MAC}(s, R_{mac})\\ (\tilde{s}, \tilde{\sigma}) \leftarrow \mathcal{A}(i, s, \sigma, R_{out})\\ \Delta := w' - w,\ \tilde{R}_{mac} := R_{mac} + g(\Delta, s, \tilde{s}, i)\end{array}\right]$$

On the right hand side of the inequality, the pair (w, w') and the value i determine the values $\Delta, s, R_{mac}, R_{out}$. But the distributions $(\Delta, s, i, R_{mac}, R_{out})$ and (Δ, s, i, U_ℓ) can be distinguished with probability at most ε, by the security of the extractor and the fact that $\hat{m} \leq \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), \Delta)$.

Hence we have:

$$\begin{aligned}&\Pr[\mathcal{A}\text{ succeeds}]\\ &\leq \varepsilon + \Pr\left[\begin{array}{c}\mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma}\\ \\ (\tilde{s}, \tilde{\sigma}) \neq (s, \sigma)\end{array}\middle|\begin{array}{c}i \leftarrow U_d,\ R_{mac} \leftarrow U_k,\ (w, w') \leftarrow (W, W')\\ s := \mathsf{SS}(w),\ \sigma := \mathsf{MAC}(s, R_{mac})\\ (\tilde{s}, \tilde{\sigma}) \leftarrow \mathcal{A}(i, s, \sigma, U_{\ell-k})\\ \Delta \leftarrow w' - w, \tilde{R}_{mac} := R_{mac} + g(\Delta, s, \tilde{s}, i)\end{array}\right]\\ &\leq \varepsilon \ + \max_{R^\Delta_{mac}, \tilde{s} \neq s, \sigma, \tilde{\sigma}} \Pr\left[\mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma}\middle|\begin{array}{c}R_{mac} \leftarrow U_k\\ \sigma := \mathsf{MAC}(s, R_{mac})\\ \tilde{R}_{mac} := R_{mac} + R^\Delta_{mac}\end{array}\right]\\ &\leq \varepsilon + \delta\end{aligned}$$

Where the last inequality follows from the security of the KMS-MAC. □

The above theorem is stated with generality in mind. We now examine the parameters we get when plugging in the optimal implementation of a KMS-MAC and using the "multiplication" extractor $\mathsf{Ext}(x, i) \stackrel{\text{def}}{=} [x \cdot i]_1^v$. Recall, we let u denote the length of the secure sketch and n denotes the length of the secret w. We define $m' = \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W)) \geq m - u$ to be the residual min entropy "left" in w after seing s. Using Theorem 5 and some simple manipulation, we finally get the following concrete corollary.

Corollary 3. *Let m, t, ε and $\delta \geq \varepsilon$ be chosen arbitrarily. Let $\rho = 2\log\left(\frac{2(u+3)}{\varepsilon(\delta-\varepsilon)}\right) + 2$. Let $v = t\left(\log\left(\frac{n}{t}\right) + \log e\right)$ be the upper bound on the volume of the Hamming ball of radius t. We construct an $(m, \ell, t, \varepsilon, \delta)$-robust fuzzy where the extracted key length ℓ is given by:*

- *For the family $\mathcal{F}^{all}_{(t,m)}$ and post-application robustness $\ell = m' - v - \rho$.*
- *For the family $\mathcal{F}^{all}_{(t,m)}$ and pre-application robustness $\ell = m' - \rho$ as long as $m' - v \geq \rho$.*
- *For the family $\mathcal{F}^{indep}_{(t,m)}$ and both pre/post-application robustness $\ell = m' - \rho$.*

The corollary is proven by bounding the value $\widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), W - W')$ for the two families $\mathcal{F}^{indep}_{(t,m)}$ and $\mathcal{F}^{all}_{(t,m)}$. We give a detailed proof in the full version of this work [8].

COMPARISON WITH PREVIOUS CONSTRUCTIONS: Recall that the "non-robust" construction of [10] extracts $\ell \leq m' - 2\log\left(\frac{1}{\varepsilon}\right)$ bits. On the other hand, the robust construction of [11] requires $\ell \leq \frac{1}{3}\left(2m - n - u - 2t\log\left(\frac{en}{t}\right) - 2\log\left(\frac{n}{\varepsilon^2\delta}\right)\right) - O(1)$. The bounds achieved in this paper are significantly closer to the non-robust version.

5.4 Extension to Other Metrics

We note that the above construction can be extended for other metric spaces and secure sketches. For example, we can easily extend our discussion of the hamming distance over a binary alphabet to an alphabet of size q where $\mathbb{F}_q$ is a field. In addition, our construction extends to the set difference metric in exactly the same way as the construction of [11].

References

1. Brassard, G., Broadbent, A., Fitzsimons, J., Gambs, S., Tapp, A.: Anonymous quantum communication. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, Springer, Heidelberg (2007)
2. Broadbent, A., Tapp, A.: Information-theoretic security without an honest majority. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, Springer, Heidelberg (2007)
3. Blundo, C., De Santis, A.: Lower bounds for robust secret sharing schemes. Information Processing Letters 63(6) (1997)
4. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)
5. Boyen, X.: Reusable cryptographic fuzzy extractors. In: 11th ACM Conference on Computer and Communication Security, ACM Press, New York (2004)
6. Cabello, S., Padró, C., Sáez, G.: Secret sharing schemes with detection of cheaters for a general access structure. Designs, Codes and Cryptography 25, 175–188 (2002); In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 175–188. Springer, Heidelberg (1997)
7. Cramer, R., Damgård, I.B., Fehr, S.: On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, Springer, Heidelberg (2001)

8. Cramer, R., Dodis, Y., Fehr, S., Padró, C. Wichs, D.: Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors. Technical Reports 2008/030, Cryptology ePrint archive, http://eprint.iacr.org/2008/030
9. Dodis, Y.: Exposure Resillient Cryptography. Ph.D. Thesis, MIT (2000)
10. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Technical Report 2003/235, Cryptology ePrint archive, Previous version appeared at EUROCRYPT 2004, http://eprint.iacr.org/2003/235
11. Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
12. Dodis, Y., Spencer, J.: On the (non-)universality of the one-time pad. In: 43rd Annual Symposium on Foundations of Computer Science, pp. 376–385. IEEE, Los Alamitos (2002)
13. Desmedt, Y., Wang, Y.: Perfectly secure message transmission revisited. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, Springer, Heidelberg (1993)
14. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. Journal of the ACM 40(1) (1993)
15. Karchmer, M., Wigderson, A.: On span programs. In: 8th Annual Conference on Structure in Complexity Theory (SCTC 1993), IEEE, Los Alamitos (1993)
16. Krawczyk, H.: Distributed fingerprints and secure information dispersal. In: 12th ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, New York (1993)
17. Nisan, N., Zuckerman, D.: Randomness is linear in space. Journal of Computer and System Sciences 52(1), 43–53 (1996)
18. Obana, S., Araki, T.: Almost Optimum Secret Sharing Schemes Secure Against Cheating for Arbitrary Secret Distribution. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, Springer, Heidelberg (2006)
19. Ogata, W., Kurosawa, K.: Optimum secret sharing scheme secure against cheating. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, Springer, Heidelberg (1996)
20. Ogata, W., Kurosawa, K., Stinson, D.R., Saido, H.: New combinatorial designs and their applications to authentication codes and secret sharing schemes. Discrete Mathematics 279, 383–405 (2004)
21. Padró, C., Sáez, G., Villar, J.L.: Detection of cheaters in vector space secret sharing schemes. Designs, Codes and Cryptography 16, 75–85 (1999)
22. Padró, C.: Robust vector space secret sharing schemes. Information Processing Letters 68, 107–111 (1998)
23. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of the ACM 36(2) (1989)
24. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery 22(11) (1979)
25. Simmons, G.J.: Authentication theory/Coding Theory. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, Springer, Heidelberg (1985)
26. Tompa, M., Woll, H.: How to share a secret with cheaters. Journal of Cryptology 1(3) (1988)

Obfuscating Point Functions with Multibit Output*

Ran Canetti[1] and Ronny Ramzi Dakdouk[2]

[1] IBM T. J. Watson Research Center, Hawthorne, NY
canetti@csail.mit.edu
[2] Yale University, New Haven, CT
dakdouk@cs.yale.edu

Abstract. We construct obfuscators of point functions with multibit output and other related functions. A point function with multibit output returns a fixed string on a single input point and zero everywhere else. Obfuscation of such functions has a useful application as a strong form of symmetric encryption which guarantees security even when the key has very low entropy: Essentially, learning information about the plaintext is paramount to finding the key via exhaustive search on the key space.

Although the constructions appear to be simple and modular, their analysis turns out to be quite intricate. In particular, we uncover some weaknesses in the current definitions of obfuscation. One weakness is that current definitions do not guarantee security even under very weak forms of composition. We thus define a notion of obfuscation that is preserved under an appropriate composition operation. The constructions can use any obfuscator of point functions under the proposed definition. Alternatively, they can use perfect one way (POW) functions with statistical indistinguishability, or with computational indistinguishability at the price of somewhat weaker security.

Keywords: obfuscation, composable obfuscation, multibit point function obfuscation, digital locker, point function obfuscation.

1 Introduction

Program Obfuscation is one of the most intriguing open problems in cryptography. Informally, a program obfuscator (or, simply, an obfuscator) is a compiler that converts a program into another one, called the obfuscated program or code, that has a similar functionality but satisfies certain secrecy requirements. Informally, the secrecy requirement stipulates that whatever "useful" information the obfuscated code reveals is learnable from the program's input/output behavior. In other words, an obfuscated program should not reveal anything useful beyond what's learned by inspecting the program's outputs on inputs of choice. This requirement is formalized by Barak *et al.* [2] through a simulation-based definition called the virtual-blackbox property. The virtual-blackbox property says that every adversary has a corresponding simulator that emulates the output of the adversary given only oracle (i.e., blackbox) access to the same functionality being obfuscated.

* Work supported by NSF grants 0331548 and CFF-0635297, and BSF grant 2006317.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 489–508, 2008.

In the same work, Barak *et al.* provide impossibility results regarding general obfuscation, even when the output of the adversary is restricted to predicates. In other words, it is shown that there are certain functionalities and corresponding predicates where these predicates are learnable from any program implementing the functionalities but not so given blackbox access to them. In light of this general negative result, we are forced to study obfuscation of restricted classes of functions if we wish to adopt the definition of [2]. Here, we follow this line of work. In particular, we build on the previous work on point function obfuscation [4,5,13,11] towards obfuscating slightly more complex functions, namely point functions with multibit output. Moreover, we show that obfuscation of point functions are not necessarily secure even under self-composition, a property needed in our analysis. We next go into a more detailed exposition of our work.

Obfuscation of point functions with multibit output. A point function returns 1 on a single input and 0 everywhere else. Formally, $F_x(y) = 1$ if $y = x$ and 0 otherwise. A point function with multibit output generalizes point functions in that it outputs, on a single input, a long string instead of 1. Formally, $F_{x,y}(z) = y$ if $z = x$, and 0 otherwise.

The connection to symmetric encryption. Obfuscators for point functions with multibit output have a useful application as what we call a **digital locker**. A digital locker is a strong form of symmetric encryption which provides meaningful security even the key is taken from a distribution with very low entropy. More specifically, the guarantee is that the complexity of learning anything about the plaintext corresponds to that of finding the key via exhaustive search over the key space. We formalize this privacy notion using the simulation paradigm in a way similar to obfuscation. Namely, we require that the behavior of the adversary on an encryption of message m with key k be simulatable given blackbox access to the multibit point function, $F_{k,m}$. Consequently, obfuscation of point functions with multibit output can be used to realize digital lockers as follows: to encrypt a message m using a key k, simply output an obfuscation of $F_{k,m}$.

Real life applications of digital lockers include password-based encryption where the human-generated password is far from uniform. For instance, Firefox has a password manager that acts as a digital locker [1]. The password manager locks website credentials using a master password chosen by the user. Then, the user has to provide this password in order to unlock the content. It is stressed that the goal here is not to prevent exhaustive search over the keys, but rather to guarantee that this is essentially the only possible attack.

The construction. Even though obfuscation of point functions with multibit output is known in the Random Oracle Model [11], it is not known in the standard model except when the function is drawn from a uniform distribution (specifically, when x in $F_{x,y}$ is uniform) [7] or when the output length of the function is short (specifically, when $|y| = O(log|x|)$) [13]. Here, we provide a transformation from point function obfuscators to obfuscators of point functions with multibit output. The idea is simple. The obfuscation of multibit point functions consists of some number of copies of obfuscated point functions. These copies have the property that the first and the ith copy correspond to an obfuscation of the same point function if and only if the ith bit in the multibit output is 1. In more detail, let $F_{a,b}$ be the multibit point function to be obfuscated, $t = |b|$, and $O(F_a, r)$ be the obfuscation of the point function, F_a, using randomness r. Then, the obfuscation of $F_{a,b}$ consists of $O(F_a, r_0), O(x_1, r_1), ...,$ $O(x_t, r_t)$, where x_i is F_a if $b_i = 1$ and x_i is a uniformly chosen point function otherwise. To recover b from the correct a and this obfuscation, first verify that $O(F_a, r_0)$ $(a) = 1$, then $b = O(x_1, r_1)(a), ..., O(x_t, r_t)(a)$.

On composing obfuscation. The construction described above is very simple and modular, and one expects that its proof be likewise. However, it turns out that this is not the case. To prove the security of the above transformation, we face an issue. Observe that our construction is composed of a concatenation of $t + 1$ obfuscated point functions. Thus, in order for our construction to be secure, the original obfuscation *has* to remain secure under composition. However, we show that the current definition of obfuscation does not guarantee composition. This is also the case even for composing multiple obfuscated copies of the *same* function. Interestingly, the statement still holds even if we consider obfuscation secure in the presence of auxiliary information. We emphasize that this is a fundamental point about the definition of obfuscation that is of independent interest.

In more detail, we show that there exists an obfuscation of point functions that reveals the input when it is self-composed. Specifically, we show an obfuscator, O, such that for any x, it is possible to recover x from $O(F_x, r_1), ..., O(F_x, r_{nlog(n)})$, where $n = |x|$.

Moreover, similar results holds for POW functions and POW functions secure with auxiliary information [4,5]. At a high level, a POW function can be thought of as an obfuscation of point function. See Appendix A for more details on POW functions and their relation to point function obfuscation.

In light of these negative results, we analyze the above construction using, as the underlying primitive, three different forms of composable obfuscation of point functions. First, if the underlying primitive is a composable obfuscation of point functions (as in simply-composable obfuscation of [11]), then this construction is a composable obfuscation of multibit point functions. This is actually a characterization: composable obfuscation of point functions exists if and only if that of point functions with multibit output exists. Second, we show that our construction is an obfuscation of multibit point functions if the underlying primitive is a statistically indistinguishable POW function.[1] Third, if the primitive is a computationally indistinguishable POW function, then the construction is an obfuscation provided that y in $F_{x,y}$, is "independent" of x.

Finally, we show how to generalize this construction to obfuscate set-membership predicates and functions for polynomial-sized sets. A set-membership predicate outputs 1 if the input belongs to the set and 0 otherwise, while a set-membership function outputs a string, y_i, if the input matches a set member, x_i, and 0 otherwise.

A tighter definition of obfuscation. The standard definition of obfuscation incorporates an unspecified "polynomial slack", in the sense that it allows the simulator to query its oracle an unspecified polynomial number of times, regardless of the complexity of the adversary. This translates to allowing obfuscation schemes that leak secret information

[1] To be accurate, the second construction satisfies approximate functionality only computationally, i.e., it is hard to efficiently find an input point on which the obfuscated function differs from the original one.

on some unspecified polynomial number of functions in a given family. In the context of digital lockers, this allows encryption schemes that, say, reveal the plaintext on a polynomial number of keys. We propose ways to fix this weakness in the definition and constructions of obfuscators and digital lockers; however our solution here is far from being completely satisfactory.

1.1 Related Work

Obfuscating Point Functions in the Random Oracle Model. Lynn *et al.* [11], inspired by the password-hiding scheme in Unix that stores a hash of the password instead of the password itself, propose a similar obfuscation of point functions in the random oracle model. In this model, an obfuscator, O, has oracle access to a truly random function, R. In order to construct an obfuscation of a point function, F_x, O queries R on x to get $z = R(x)$ and then stores z in the obfuscated code, $O(F_x)$. $O(F_x)$ also contains preprocessing code which on input y returns 1 if and only if $R(y) = z$.

It is easy to see that $O(F_x)$ and F_x have approximate functionality (they have the same functionality almost always). Intuitively, $O(F_x)$ is an obfuscation of F_x because R's answers on queries are completely independent and random. So, storing $R(x)$ does not reveal any information about x, but it allows verification of a guess, which is also achievable via oracle access to F_x.

Also, Lynn *et al.* [11] generalize this construction to obfuscate multibit output point functions and set-membership predicates and functions in the random oracle model. To obfuscate a multibit point function, $F_{x,y}$, choose a random r, and output $r, R_1(x,r)$, $R_2(x,r) \oplus y$, where R_1 and R_2 denote the first and second half of the bits of $R(.)$. This construction is secure under composition (as in Definition 2 or the simply-composable definition of [11]). In Section 3.2, we instantiate this scheme. The resulting construction is more efficient than our first one but uses a stronger assumption.

Obfuscating Point Functions in the standard model. Perfectly one-way (POW) functions [4] can be used to obfuscate a point function F_x by replacing the random oracle in [11] with a POW function, H. Here, instead of storing $R(x)$, we store $H(x)$ in the obfuscated code and use the verifier for H to determine if $H(x)$ is a valid hash of the input.

Canetti [4] constructs a POW hash function based on a strong version of the Diffie-Hellman assumption. In particular, it assumes that the Diffie-Hellman assumption holds not only against uniform distributions but also with respect to any well-spread distribution. Moreover, Wee [13] shows how to obfuscate point functions and point functions with logarithmic output based on a strong one-way permutation assumption. Specifically, the assumption is that any polynomial-time machine can invert the permutation on at most a polynomial number of points. The two constructions mentioned so far (and our construction as well) use a weaker notion of obfuscation than the one in [2]. Specifically, the simulator in [4,13] depends on the simulation-error gap between the adversary and the simulator. (see Definition 1 for more detail).

Canetti *et al.* [5] provide two constructions of POW functions based on standard computational assumptions (in particular, based on either claw-free permutations or one-way permutations). The simulator for these constructions does not depend on the gap. However, the input distribution is assumed to have high min-entropy (n^ϵ). Moreover, Futoransky *et al.* [7] show how to obfuscate point functions and point functions with multibit output based on standard assumption. However, the input distribution is assumed to be uniform. Finally, Hofheinz *et al.* [10] obfuscate point functions *deterministically*. However, the secrecy requirement does not guarantee no information leakage, rather that it is hard to recover the input in its entirety. This obfuscation is self-composable because the obfuscator is deterministic. However, it is not composable according to our notion. In particular, different obfuscated point functions can not be securely composed.

Encryption with imperfect randomness. The question of encryption with "imperfect randomness" is studied also in [12,6,3], yielding some strong impossibility results. However, these results do not apply to our case since they assume that the parties have no source of perfect randomness, whereas we allow the parties to use perfect randomness other than the key. In our setting, symmetric encryption with imperfect keys can be constructed using randomness extractors in standard ways, as long as the distribution of the key has sufficient min-entropy. Here however we are concerned with the case where there is no a priori guarantee on the min-entropy of the key.

1.2 Organization

In Section 2, we recall common notations and definitions including that of obfuscation, leaving definitions of POW functions to Appendix A. We present our construction and analyze it in Section 3. (We also present a more efficient construction under a stronger assumption in Section 3.2.) In Section 4, we study the issue of composable obfuscation. Finally, we discuss the connection to encryption schemes in Section 5.

2 Preliminaries

Let X_n denote a probability distribution on $\{0,1\}^n$ and U_n the uniform distribution on $\{0,1\}^n$. Then, $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ is called a distribution ensemble (distribution for short). A distribution is called **well-spread** if it has superlogarithmic min-entropy, i.e., $max_k Pr[X_n = k]$ is a negligible function in n. Moreover, $a \leftarrow D_n$ means that a is chosen from $\{0,1\}^n$ according to distribution D_n. Also, denote by $\Delta(X_n, Y_n)$ the statistical difference between the two distributions X_n and Y_n over $\{0,1\}^n$. Formally, $\Delta(X_n, Y_n) = \frac{1}{2}\Sigma_{a\in\{0,1\}^n}|Pr[X_n = a] - Pr[Y_n = a]|$.

A probabilistic function family is a set of efficient probabilistic functions having common input and output domains. Formally, $\mathbf{H}^n = \{H_k\}_{k\in K_n}$ is a function family with key space K_n and randomness domain R_n if, for all $k \in K_n, H_k : I_n \times R_n \to O_n$. A probabilistic function family has **public randomness** if for all k, $H_k(x,r) = r, H'_k(x,r)$ for some deterministic function H'_k. A family ensemble is a collection of function families, i.e., $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$.

Let PPT denote any probabilistic polynomial-time Turing machine, and nonuniform PPT any probabilistic polynomial-sized circuit family. A PPT (respectively nonuniform PPT) A with oracle access to O is denoted by A^O.

A function, μ, is called negligible if it decreases faster than any inverse polynomial. Formally, it is negligible if, for any polynomial p, there exists an N_p such that, for all $n \geq N_p$: $\mu(n) < \frac{1}{p(n)}$. In this work, we reserve μ to denote negligible functions. An uninvertible function, f, with respect to a well-spread distribution, $\mathbb{X}$, is an efficiently computable function that is hard to invert on $\mathbb{X}$. Formally, for any PPT, A, $Pr[x \leftarrow X_n,\ A(f(x)) = x] < \mu(n)$.

A **set-membership predicate**, $F_{S=\{x_1,...,x_t\}} : \{0,1\}^n \rightarrow \{0,1\}$, outputs 1 if and only if its input is in S. Here, S is assumed to have at most polynomially many elements. A **set-membership function**, $F_{(x_1,y_1),...,(x_t,y_t)} : \{0,1\}^n \rightarrow \{y_1, ..., y_t, 0\}$ outputs y_i if and only if the input matches x_i.

2.1 Obfuscation

We adopt the definition of obfuscation used in [4,13] because obfuscation of point functions is known for this notion only (if the distribution on this class of functions is not restricted). This definition is weaker than the one in [2] because the size of the simulator is allowed to depend on the quality of the simulation. Formally,

Definition 1 (Obfuscation). *Let $\mathbb{F}$ be any family of functions. A PPT, O, is called an* ***obfuscator*** *of $\mathbb{F}$, if:*

1. ***Approximate Functionality*** *For any $F \in \mathbb{F}$: $Pr[\exists x,\ O(F)(x) \neq F(x)]$ is negligible. Here, the probability is taken over the coin tosses of O.*
2. ***Polynomial Slowdown*** *There is a polynomial p such that, for any $F \in \mathbb{F}$, $O(F)$ runs in time at most $p(T_F)$, where T_F is the worst-case running time of F.*
3. ***Weak Virtual Black-box Property*** *For any nonuniform PPT A and any polynomial p, there exists a nonuniform PPT S such that for any $F \in \mathbb{F}$ and sufficiently large n:*
$$|Pr[b \leftarrow A(O(F)) : b = 1] - Pr[b \leftarrow S^F(1^{|F|}) : b = 1]| \leq \frac{1}{p(n)}.$$

3 Obfuscating Point Functions with Multibit Output

We show how to obfuscate point functions with multibit output as well as set-membership predicates and functions for polynomial-sized sets. Because the constructions and proofs for obfuscating set-membership predicates and functions are similar to that for multibit point function, we focus on the latter. We comment on the former in Section 3.1. We also present a more efficient obfuscation of multibit point functions using a stronger assumption in Section 3.2.

We use obfuscated point functions as building blocks in obfuscating point functions with multibit output. The idea is simple. To obfuscate $F_{x,y}$, we encode y bit-by-bit using an obfuscator for F_x. Specifically, if the ith bit of y is 1, it is encoded as an obfuscation of F_x, otherwise, it is encoded as an obfuscation of an independent and uniform point function. In more detail, let H be a randomized obfuscator for point functions. Then the obfuscation contains $H(F_x, r), H(F_{x_1}, r_1), ..., H(F_{x_t}, r_t)$, where $t = |y|$ and $x_i = x$ if the ith bit of y is 1, otherwise, x_i is uniform. The first obfuscated point functions always corresponds to x, and is used to check whether the input is actually x. Now, y can be recovered given $z = x$. First, check that $H(F_x, r)(z) = 1$. If so, for every i, $y_i = H(F_{x_i}, r_i)(z)$.

Formally, we present an obfuscator, O, for the class of multibit output point functions, $\mathbb{F}$. O, on input $F_{x,y}$, where y has length t, selects $r_1, ..., r_{t+1}$ from R_n, the randomness domain of the point function obfuscator, H. It then computes $H(F_x, r_1)$. It also computes $H(F_x, r_{i+1})$ if the $y_i = 1$ and $H(z_{i+1}, r_{i+1})$ otherwise, where z_{i+1} is uniform. Let $u_x = u_1, ..., u_{t+1}$ be the sequence of obfuscated functions just computed. Then O outputs the following obfuscation, $O(F_{x,y})$, with u_x stored in it.

```
input: a
if u_1(a) = 0 then
    return 0;
else
    for i ← 2 to t + 1 do
        if u_i(a) = 1 then
            y_{i-1} ← 1;
        else
            y_{i-1} ← 0;
        return y = y_1, ..., y_t;
    end
```

Algorithm 1. $O(F_{x,y})$

Analysis. This construction is simple and modular. It is possible to replace H by any relative of point function obfuscation such as POW functions (see Appendix A) and analyze the security of the construction based on the security of the underlying primitive. We would like to prove that our construction is secure based on the simple assumption that the underlying primitive is an obfuscation of point functions. However, as we show in Section 4, this is not possible. This is so because the definition of obfuscation does not guarantee even self-composition. Thus, there exist point function obfuscators and POW functions for which this construction is provably insecure.

We investigate the secrecy of this construction based on three underlying primitives with different composition properties. In the first case, we consider the notion of composable obfuscation (as in Definition 2, also known as simply-composable obfuscation in [11]). We show a characterization that composable point function obfuscation exists if and only if composable multibit point function obfuscation exists. In the second case, we show that if H is a statistically indistinguishable POW function, then our construction is secure. Finally, if H is a computationally indistinguishable POW then this construction satisfies a weaker form of obfuscation where y, in $F_{x,y}$, has to be independent of x.

Analysis based on composable obfuscation. In this work, composable obfuscation refers to the fact that concatenating any sequence of obfuscated functions, where the functions are taken from the same class, constitutes an obfuscation for that sequence of functions. This form of composition, also known as simply-composable obfuscation in [11], should not be confused with self-composition which means that concatenating a sequence of independent obfuscation of the same function does not compromise secrecy. Formally,

Definition 2 **(t-Composable Obfuscation, [11]).** *Let $\mathbb{F}$ be any family of functions. A PPT, O, is called a t-composable obfuscator for $\mathbb{F}$, if:*

1. *Approximate functionality and polynomial slowdown are as before.*
2. ***Virtual Black-box property** For any nonuniform PPT, A, and any polynomial, p, there is a nonuniform PPT, S, such that for any functions $F_1, ..., F_{t(n)} \in \mathbb{F}$ (n is a security parameters, e.g., $n = |F_1| = ... = |F_{t(n)}|$) and sufficiently large n:*

$$|Pr[b \leftarrow A(O(F_1), ...O(F_{t(n)}) : b = 1] - Pr[b \leftarrow S^{F_1,...,F_{t(n)}}(1^n) : b = 1]| \leq \frac{1}{p(n)}$$

If O is a t-composable obfuscator for $\mathbb{F}$ for any polynomial t, then it is called a composable obfuscator.

If H satisfies $(t+1)$-composable obfuscation for some t, then our construction can be shown to be an obfuscation of multibit point function with output length t. Approximate functionality and polynomial slowdown follow from the corresponding properties on H. By the virtual black-box property on H, the output of $A(O(F_{x,y}) = O(F_x), O(F_{x_1}), ..., O(F_{x_{t(n)}}))$ can be simulated by $S^{F_x, F_{x_1}, ..., F_{x_{t(n)}}}(1^n)$, where $x_i = F_x$ if $y_i = 1$ and x_i is uniform otherwise. Moreover, oracle access to $F_x, F_{x_1}, ..., F_{x_{t(n)}}$ can be simulated with oracle access to $F_{x,y}$: If S queries any of its oracle on a point z such that $F_{x,y}(z) = 0$, then answer 0 (this may incur a negligible simulation error only), otherwise, $z = x$ so y can be fully recovered. Thus, this construction satisfies the virtual black-box property.

Observe that our construction is a *composable* obfuscation of multibit point functions with the appropriate parameters. Specifically, if the output length of the multibit point function is restricted to at most t, then this construction is a t'-composable obfuscation if H is $(t+1)t'$-composable. In addition, it is easy to see that the existence of a t-composable obfuscation of multibit point functions implies a t-composable obfuscation of point functions. Formally, we have the following characterization with a proof that follows the above discussion.

Theorem 1. *Composable obfuscators of point functions with multibit output exist if and only if composable obfuscators of point functions exist.*

Specifically, if a point function obfuscator, H, is $(t+1)t'$-composable (as in Definition 2) then the above construction is a t'-composable obfuscation of multibit point functions with output length t. On the other hand, a t-composable obfuscation of multibit point functions implies a t-composable obfuscation of point functions.

Analysis based on statistical indistinguishability. Suppose $\mathbb{G}$ is a statistically indistinguishable POW family ensemble (see Appendix A for the formal definition). We can replace H by $\mathbb{G}$ in the above construction. Specifically, the obfuscator, O, samples a key, k, for $\mathbb{G}$ and replaces $H(x, .)(a)$ with $V(a, G_k(x, .))$, where V is the verification algorithm for $\mathbb{G}$. This results in an obfuscation of point function with multibit output except with *computational approximate functionality* [13], i.e, no adversary can efficiently find a point on which the original function differs from the obfuscated one. This relaxation to approximate functionality is necessary when using statistical POW functions because they can not be statistically collision resistant. On the other hand, we argue that the result satisfies the virtual-blackbox property. Informally, from the fact that $\mathbb{G}$ is a statistical POW function we can conclude that an obfuscation of $F_{x,y}$, where x is taken from a well-spread distribution and y is arbitrary, is statistically close to a sequence of hashes of random inputs. It follows that for all but polynomially many x, an obfuscation of $F_{x,y}$ is indistinguishable from random hashes. Consequently, we get a simulator that runs the adversary on random hashes unless x is taken from that polynomial set, in which case the simulator can recover y and run the adversary on an obfuscation of $F_{x,y}$. Formally,

Theorem 2. *Let $\mathbb{G}$ be a statistically $(t+1)$-indistinguishable POW function (as in Definition 8). Then, the above construction is an obfuscation of point functions with multibit output length t (as in Definition 1), where approximate functionality is only computational.*

Proof (Sketch). Polynomial slowdown follows immediately from the fact that $\mathbb{G}$ has a polynomial output length. Also, by public verification and collision resistance of POW functions (definition 6), it follows that O satisfies computational approximate functionality.

Virtual black-box property. Recall, the definition of statistical indistinguishability says that for any well-spread distribution, $\mathbb{X}$:

$$\Delta(G_k(X_n, R_n^1), ..., G_k(X_n, R_n^{(t+1)(n)}), G_k(U_n^1, R_n^1), ..., G_k(U_n^{t(n)}, R_n^{(t+1)(n)})$$

is negligible, where each distribution R_n^i (respectively, U_n^i) is the same as R_n (respectively, U_n).

Using the fact that for any function, λ, $\Delta(\lambda(X), \lambda(Y)) \leq \Delta(X, Y)$, we have for any distribution,$\mathbb{XY}$ on (x, y), where the corresponding distribution on x is well-spread:

$$\Delta(O(F_{X_n,Y_n}), G_k(U_n^1, R_n^1), ..., G_k(U_n^{t(n)}, R_n^{(t+1)(n)}) \quad (1)$$

is negligible. (We assume without loss of generality that $O(F_{x,y})$ consists only of the $t+1$ $\mathbb{G}$-hashes.)

Using the same technique from the proof of Theorem 4 in [4], it can be shown that $O(F_{x,y})$ is indistinguishable from $\mathbb{G}$-hashes of uniform strings on all but a polynomial number of x. That is, for any nonuniform PPT, A, and any polynomial, p, there exists a polynomial size family of sets, $\{L_n\}$, such that for sufficiently large n, and $x \notin L_n$ and any y:

$$|Pr[b \leftarrow A(O(F_{x,y})) : b = 1] - Pr[u_1, ..., u_{t+1} \leftarrow U_n, ..., U_n,$$

$$r_1, ..., r_{t+1} \leftarrow R_n, ..., R_n,\ b \leftarrow A(G_k(u_1, r_1), ..., G_k(u_{t+1}, r_{t+1})) : b = 1]| \leq \frac{1}{p(n)}. \quad (2)$$

Intuitively, this is true because otherwise there is a super-polynomial number of values for x (with a corresponding value for y), on which A can distinguish $O(F_{x,y})$ from hashes of random strings. By defining a well-spread distribution, e.g., a uniform distribution, on this superpolynomial number of values for x, A violates (1).

Now, for any nonuniform PPT, A, and a polynomial, p, we construct a nonuniform PPT, S that simulates A. S receives the polynomial-size set, L_n, as an advice string. It checks if the oracle, $F_{x,y}$, responds with the nonzero value, y, to any element in the set, L_n. If so, then S can compute $O(F_{x,y})$ and simulate A on it. Otherwise, x is not in L_n, so S runs A on hashes of random inputs. By (2), this is close to a true simulation. For more detail, we refer the reader to the proof of Theorem 4 in [4]. □

Analysis based on computational indistinguishability. We would like to weaken the assumption in Theorem 2 to computational indistinguishability. However, it is not clear how to use computational indistinguishability, i.e., $G_k(x, r_1), ..., G_k(x, r_{t+1})$ is computationally indistinguishable from hashes of uniform, to conclude that $O(F_{x,y})$ is indistinguishable from hashes of random inputs. It seems that the problem lies in the potential dependence of y on x, e.g., y may be equal to x. This is not a problem in the statistical case because we can use the fact that statistical difference does not increase by applying the same function on both distributions. In the computational setting, if we use the traditional blackbox reduction, we need to construct $O(F_{x,y})$ from hashes of x and then run A on it. However, it is not clear how to do so if $y = x$. On the other hand, suppose y is independent of x, e.g., y is taken independently from a uniform distribution. Then, for some y, it is possible to compute $O(F_{x,y})$ given hashes of x, $G_k(x, r_1), ..., G_k(x, r_{t+1})$, by replacing $G_k(x, r_i)$ with a hash of a random string if the ith bit of y is 0. Thus, we know that computational indistinguishability gives us a weaker notion of obfuscation where the simulator depends on the distribution on y. Whether computational indistinguishability gives us the standard virtual-blackbox property remains unknown. Nevertheless, this weak obfuscation can be used as a digital locker as described in the introduction. The caveat is that the message being encrypted should be independent of the encryption key. This is the case if, for instance, the message is chosen without knowledge of the key.

Formally, the virtual black-box property becomes: for any nonuniform PPT A, any polynomial p, and any (efficiently samplable) distribution $\mathbb{Y}$, there exists a nonuniform PPT S such that for any x and sufficiently large n:

$$|Pr[y \leftarrow Y_n,\ b \leftarrow A(O(F_{x,y})) : b = 1] - Pr[y \leftarrow Y_n,\ b \leftarrow S^{F_{x,y}}(1^{|F_{x,y}|}) : b = 1]| \leq \frac{1}{p(n)}. \quad (3)$$

Also, we remark that this construction has either approximate or computational approximate functionality depending on whether the POW function satisfies statistical or computational collision resistance. Formally, we have the following theorem whose proof follows that of Theorem 2 and the above discussion, and is not recreated here.

Theorem 3. *If $\mathbb{G}$ is a computationally $(t + 1)$-indistinguishable POW function, then the above construction is an obfuscation of point function with output length t, where the virtual-blackbox property is as in (3).*

3.1 Obfuscating Set-Membership Predicates and Functions

To obfuscate a set-membership predicate, simply obfuscate the point functions on every element in the set (this is feasible because the set has a polynomial size), and then store all the obfuscated functions in a randomly permuted order. To determine whether a particular input is in the set, we only need to check whether any of the obfuscated functions outputs 1 on this input. It can be shown that this construction is an obfuscation of set-membership predicate based on composable obfuscation of point functions. Moreover, to obfuscate a set-membership function, $F_{(x_1,y_1),\ldots,(x_t,y_t)}$, we only need to run the obfuscator for the multibit output point function on each F_{x_i,y_i}, and then store these obfuscated functions in a randomly permuted order. It can be shown that composable obfuscation of point functions is a necessary and sufficient condition for the security of this construction.

3.2 A More Efficient Obfuscation of Point Functions with Multibit Output

We note that the obfuscation of point function with multibit output in the RO model [11] can be instantiated by using a stronger assumption on the underlying primitive. The end result is a more efficient construction than the one described previously. Specifically, let $\mathbb{G}$ be a POW function with public randomness. To obfuscate $F_{x,y}$, select r_1 and r_2 uniformly from the randomness domain of $\mathbb{G}$ and output $G_k(x, r_1), r_2, z$, where $G_k(x, r_2) = (r_2, v)$ and $z = y \oplus v$.[2] To recover y from (a, b, c) and x', first check that $V(x', a) = 1$, if so, then return $y = c \oplus v$, where $G_k(x', b) = (b, v)$. Even though this construction is more efficient than the first one, it suffers from two problems. First, in order to completely hide y, it is not sufficient that $\mathbb{G}$ be indistinguishable as in Definition 9 rather its output has to be *indistinguishable from uniform*. If, for example, the first bit of the hash is always 0, then the first bit of y is revealed. Second, for the proof to go through, we need to assume that $\mathbb{G}$ is *statistically* indistinguishable from *uniform* because y may depend on x. Contrast this assumption with the one used in Theorem 2, where $\mathbb{G}$ is statistically indistinguishable from *hashes of uniform strings*.

4 On Composable Obfuscation of Point Functions

In Section 3, we provided a transformation from an obfuscation of a point function to an obfuscation of a point function with multibit output. This transformation requires an essential property on the given obfuscation, specifically, composition. In other words, our construction assumes that we have an obfuscation of a point function such that security is not compromised when multiple obfuscated functions are given. Notably, Theorems 1, 2, and 3 all assume that H satisfies some form of composable security. Since the obfuscator is probabilistic, composable security is nontrivial. In this section, we address this question. Specifically, does the basic definition of obfuscation imply composition? From a different angle, Canetti *et al.* [5] ask if semantic perfect one-wayness implies indistinguishable perfect one-wayness or if t-indistinguishable POW functions

[2] Without loss of generality, we assume that y and v have the same length. Otherwise, the input should be of a longer input, say $x0^t$.

are $t+1$-indistinguishable. We answer these questions negatively: such primitives are not necessarily secure even under self-composition.[3] In more detail, we show that weak c-indistinguishable POW functions (where the probability is taken over the choices of the seed as well, [5]) are not necessarily $c+1$-indistinguishable for any constant c. We also show that POW functions, POW functions with auxiliary input, and obfuscation of point functions do not imply composition. Specifically, 1-indistinguishable POW functions and obfuscation of point functions are not necessarily secure for a polynomial number of copies. Moreover, even though 1-indistinguishable POW functions with auxiliary input is also c-indistinguishable for any constant c, it is not necessarily t-indistinguishable with auxiliary input for a polylogarithmic t.

In Section 4.1, we show a tight impossibility result for weak POW functions. Specifically, we show that for any constant c, weak c-indistinguishable POW functions are not weakly $c+1$-indistinguishable. We also show that if t is polynomial, then weak t-indistinguishable POW functions are not weakly $n(t+1)^2$-indistinguishable. In Section 4.2, we prove that sematic POW functions, 1-indistinguishable POW functions, and point function obfuscation are not secure if composed roughly $nlog(n)$ times. Moreover, if we consider the same functions with respect to auxiliary information, then we have a tighter result where they are not secure with respect to auxiliary information if composed superlogarithmically-many times.

4.1 Weak POW Functions Are Not Self-composable in General

A weak POW function deviates from Definition 9 in that the probability is taken over the choices of the function key as well. Here, we show that a weak c-indistinguishable POW function with respect to the uniform distribution may not be $c+1$ indistinguishable for any constant c. The idea is simple: we take any weak $3c$-indistinguishable POW function and convert it into a new function that is c-indistinguishable but the output contains shares of the input such that it is easy to compute the input from $c+1$ hashes. Informally, we add c uniform strings to the original seed and make sure that a hash of the input using any one of those c strings appears in the output with probability $\frac{1}{c+1}$. Also, with the same probability the exclusive-or of the input and all the aforementioned hashes appears in the output. Therefore, if the output of the function contains all c hashes and the exclusive-or of these hashes with the input, then it is easy to recover the input.

Formally, let $\mathbb{H}$ be any (possibly weak) $3c$-indistinguishable POW function with key space, K_n, and public randomness. We also assume that $\mathbb{H}$ is also $3c$-indistinguishable from uniform. Define a new family ensemble, $\mathbb{G}$, with a key space $(K_n, \underbrace{R_n, ..., R_n}_{c})$, an input domain $(\{0,1\}^n, \{0,1\}^n)$, and randomness domain $(R_n, \{0,1\}^{logc})$, as follows:

$$G_{k,u_1,\ldots,u_c}((x_1,x_2),(r_1,r_2)) =$$
$$\begin{cases} r_2, H_k(x_1,r_1), H_k(x_2,r_1), H_k(x_1,u_{r_2}) & \text{if } r_2 \neq 0 \\ r_2, H_k(x_1,r_1), H_k(x_1,u_1) \oplus H_k(x_1,u_2) ... \oplus H_k(x_1,u_c) \oplus x_2 & \text{if } r_2 = 0 \end{cases}$$

[3] Recall, self-composition refers to concatenation of multiple outputs of a randomized function on the *same* input.

Now, observe that it is easy to recover x_2 from $G_{k,u_1,...,u_c}((x_1, x_2), (r_1^0, 0)), ...,$ $G_{k,u_1,...,u_c}((x_1, x_2), (r_1^c, c))$. Thus, $\mathbb{G}$ is not $(c + 1)$-indistinguishable because $c + 1$ randomly-chosen hashes of (x_1, x_2) have distinct r_2 (i.e., match the aforementioned hashes) with probability $\frac{(c+1)!}{(c+1)^{c+1}}$. On the other hand, we argue that $\mathbb{G}$ is a weak c-indistinguishable POW function with respect to the uniform distribution . First, completeness and collision resistance follow from that on $\mathbb{H}$. Second,

$$G_{k,u_1,...,u_c}((x_1, x_2), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((x_1, x_2), (r_1^c, r_2^c))$$

is indistinguishable from

$$G_{k,u_1,...,u_c}((v_1, x_2), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((v_c, x_2), (r_1^c, r_2^c))$$

by the $3c$-indistinguishability property on $\mathbb{H}$, where $v_1, ..., v_c$ are uniform and independent. Moreover, by the $3c$-indistinguishability from uniform, we have

$$G_{k,u_1,...,u_c}((v_1, x_2), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((v_c, x_2), (r_1^c, r_2^c))$$

is indistinguishable from

$$G_{k,u_1,...,u_c}((v_1, w_1), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((v_c, w_c), (r_1^c, r_2^c)),$$

where $w_1, ..., w_c$ are uniform and independent.

Moreover, this result can be generalized to any polynomial t. If $\mathbb{H}$ is $3t$-indistinguishable from uniform, then $\mathbb{G}$ is a weak t-indistinguishable POW function with respect to the uniform distribution. On the other hand, $\mathbb{G}$ is not $n(t + 1)^2$-indistinguishable with respect to the uniform distribution. This is so because all the $(t + 1)$ "shares" appear in $n(t + 1)^2$ hashes with overwhelming probability. This result is stated formally in the following theorem.

Theorem 4. *Let $\mathbb{H}$ be any weak POW function that is $3t$-indistinguishable from uniform and has public randomness. Then for any constant $c \leq t$, there exist weak POW functions that are c-indistinguishable (respectively, t-indistinguishable) with respect to the uniform distribution but not $c + 1$-indistinguishable (respectively, $n(t + 1)^2$-indistinguishable) with respect to the uniform distribution.*

4.2 Point Function Obfuscation and POW Functions Are Not Self-composable in General

We show that POW functions, POW functions with auxiliary input, obfuscation of point functions, and obfuscation of point functions with auxiliary input are not generally self-composable. Also, we note that the obfuscation of point functions in [13] is not self-composable as well. The idea is simple, we start with a POW function and append to its output a hardcore bit, specifically the inner product between the input and a random string. This hardcore bit does not compromise security of a single hash. However, the function becomes completely insecure for polynomially many hashes as the input can be recovered with high probability by solving a linear system of equations.

Here, we present the proof for the case of POW functions with auxiliary input only. Let $\mathbb{H}$ be a POW function that is 1-indistinguishable with auxiliary input. Define a new family ensemble, $\mathbb{G}$:

$$G_k(x, (r_1, r_2)) = r_2, H_k(x, r_1), \langle x, r_2 \rangle,$$

where $\langle x, r_2 \rangle$ is the inner product of x and r_2 mod 2. We argue that $\mathbb{G}$ is 1-indistinguishable with auxiliary input. First, completeness and collision resistance follow from that on $\mathbb{H}$. Moreover, for any uninvertible function F, $F(x), H(x, r_1), r_2$ is one-way in x because $\mathbb{H}$ is 1-indistinguishable with auxiliary input. Therefore, by Goldreich-Levin theorem [8], we have that $F(x), r_2, H(x, r_1), \langle x, r_2 \rangle$ is indistinguishable from $F(x), r_2, H(x, r_1), b$, where b is uniform. Moreover, by 1-indistinguishability with auxiliary input on $\mathbb{H}$, $F(x), r_2, H(x, r_1), b$, is indistinguishable from $F(x), r_2, H(U_n, r_1), b$.

On the other hand, $\mathbb{G}$ is not polylogarithmically indistinguishable with auxiliary input. To see that, let F be a function that outputs the last $n - \omega(1)log(n)$ bits of its input. Then, F is uninvertible with respect to the uniform distribution. However, we argue that given $F(x)$ and a polylogarithmic number of hashes, x can be recovered completely by solving a system of linear equations. Formally,

Lemma 1. *For any two constants c and ϵ, there exists a t, which is polylogarithmic in n (specifically, $t = \omega(1)log(n)log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}$) and a PPT, A, such that for any $k \in K_n$:*

$$Pr[x \leftarrow U_n,\ r_1, ..., r_t \leftarrow R_n^G, ..., R_n^G,\ A(F(x), G_k(x, r_1), ..., G_k(x, r_t))] \geq \frac{1}{n^c}.$$

Proof. Let A be a PPT that ignores all $\mathbb{H}$ hashes ($H_k(x, .)$) but plugs-in the values of the last $n - \omega(1)log(n)$ bits of x in the system of linear equations: $r_1^2, \langle x, r_1^2 \rangle, ..., r_t^2, \langle x, r_t^2 \rangle$. We show that by solving this system we can recover x with probability $\frac{1}{n^c}$. Given the last $n - \omega(1)log(n)$ bits of x revealed by F, we can recover x from $\omega(1)log(n)$ linearly independent equations on the first $\omega(1)log(n)$ bits. Thus, we need to show that we have this many linearly independent equations in t uniformly chosen equations with probability $\frac{1}{n^c}$. First, observe that a uniform and independent r is linearly independent from $\omega(1)log(n) - 1$ or less equations with probability at least $\frac{1}{2}$. Consequently, the probability that t equations contain $\omega(1)log(n)$ linearly independent equations is at least:

$$(1 - \frac{1}{2^{log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}}})^{\omega(1)log(n)} \geq e^{ln(\frac{1}{n^c}+\epsilon)} - \epsilon = \frac{1}{n^c}.$$

□

Using the same construction, $\mathbb{G}$, it is possible to show that 1-indistinguishable POW functions (respectively obfuscation of point functions) are not necessarily t-indistinguishable (respectively, secure under t-self-composition), where $t = nlog\frac{n}{-ln(\frac{1}{n^c}+\epsilon)}$. As a concrete example, the same analysis can be used to show that the obfuscation of point function in [13] is not secure when composing t obfuscated copies of the same point function. These results can be stated formally as follows.

Theorem 5. *If there exists a* 1*-indistinguishable POW function (respectively, a point function obfuscation) with auxiliary input then there exists another* 1*-indistinguishable POW function (respectively, another point function obfuscation) with auxiliary input such that for any constants* c *and* ϵ*, the latter is not* t*-indistinguishable (respectively, is not a* t*-self-composable point function obfuscation) with auxiliary input with respect to the uniform distribution , where* $t = \omega(1)log(n)log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}$.

Moreover, if there exists a 1*-indistinguishable POW function (respectively, a point function obfuscation) then there exists another* 1*-indistinguishable POW function (respectively, another point function obfuscation) such that for any constants* c *and* ϵ*, the latter is not* t*-indistinguishable (respectively, is not a* t*-self-composable point function obfuscation) with respect to the uniform distribution, where* $t = nlog\frac{n}{-ln(\frac{1}{n^c}+\epsilon)}$.

5 On the Relationship between Obfuscation of Point Functions with Multibit Output and Symmetric Encryption

It is interesting to note that obfuscation of point functions with multibit output and symmetric encryption are similar. At the conceptual level, they capture the same idea except with a subtle difference. First, both of them satisfy the same correctness property. In particular, an encryption scheme (respectively, obfuscation of point function with multibit output) allows the recovery of the message (respectively, y) given the key (respectively, x). Second, they share similar privacy requirements. An obfuscation hides the special output, y, of the function, $F_{x,y}$ unless x is given. Likewise, a symmetric encryption should ensure the privacy of the message unless the adversary possesses the key. However, the former primitive differs from the latter in that its behavior is defined over all possible input x, while the decryption scheme leaves the behavior undefined on wrong keys. In other words, one may, at least conceptually, think of an obfuscation of point functions with multibit output as a *special* form of encryption, where wrong keys are promptly detected by the decryption algorithm.

At a more technical level, another difference arises, regarding the assumption on the key distribution. Recall that symmetric encryption requires uniform keys. On the other hand, an obfuscation of point functions with multibit output does not assume anything about the distribution on x. Specifically, it provides a definition of privacy for any x. Thus, casting the former primitive as an encryption scheme ,i.e., as $O(F_{key,message})$, gives us an encryption scheme with the same privacy as defined for obfuscation. In other words, any predicate computed from the ciphertext can also be computed by exhaustively searching for the right key to recover the message. Formally,

Definition 3 (Single-message encryption for any key). *A symmetric encryption scheme,* (E, D)*, satisfies privacy for* ***any*** *key if for any nonuniform PPT* A*, and any polynomial* p*, there exists a nonuniform PPT* S *such that for any key,* k*, any message,* m*, and sufficiently large* n*:*

$$|Pr[b \leftarrow A(E(k, m)) : b = 1] - Pr[b \leftarrow S^{F_{k,m}}(1^n) : b = 1]| \leq \frac{1}{p(n)}.$$

Observe that in the special case where the key is uniform or even sampled from a well-spread distribution, Definition 3 implies that whatever predicate computed from the ciphertext can be computed *without it (and without oracle access to $F_{k,m}$)*. Formally, an encryption scheme satisfying Definition 3 also satisfies the following privacy property.

Definition 4 (Single-message encryption with well-spread keys). *A symmetric encryption scheme, (E, D), satisfies privacy for* ***well-spread*** *keys if for any nonuniform PPT A, and any polynomial p, there exists a nonuniform PPT S such that for any well-spread distribution, $\mathbb{K} = \{K_n\}_{n \in \mathbb{N}}$, any message m, and sufficiently large n:*

$$|Pr[k \leftarrow K_n,\ b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S(1^n) : b = 1]| \leq \frac{1}{p(n)}.$$

Although Definitions 3 and 4 consider single-message encryption, encryption of multiple messages can be readily achieved using appropriately composable obfuscation of point functions with multibit output.

5.1 Weakness of Definition 3

It may seem that Definition 3 captures our intuition that the only way of breaking the encryption scheme is through exhaustively searching for the correct key. However, it turns out that this definition is not strong enough. Specifically, there are encryption schemes that satisfy this definition but reveal the plaintext when the key is taken from a polynomial-size set. For instance, modify any encryption scheme that satisfies Definition 3 so that it reveals the plaintext when the key is one of the first n lexicographically-ordered keys. The new scheme still satisfies this definition because the simulator can query the oracle on those n keys to recover the message. However, this scheme does not match our intuitive requirement. This is so because an adversary can, in constant time, output the first bit of the plaintext on the first n keys but the simulator needs $O(n)$ time to do the same. We stress that this weakness is already inherent in the notion of obfuscation, not just in the application to encryption.

Coming up with a realizable definition that captures our intuition about encryption with low-entropy keys is interesting, given the potential applications in password-based encryption. Here, we take a step in this direction. We strengthen Definition 3 by restricting the number of queries of the simulator to some fixed polynomial in the running time of the adversary and the simulation error. In more detail, for any key, k, the number of queries the simulator makes in the worst case is bounded by a fixed polynomial in the worst-case running-time of the adversary, and the simulation error.

Definition 5 (t-secure encryption). *A symmetric encryption scheme, (E, D), is t-secure if for any nonuniform PPT A, and any polynomial p, there exists a nonuniform PPT S such that for any key, k, any message, m, and sufficiently large n:*

$$|Pr[b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S^{F_{k,m}}(1^n) : b = 1]| \leq \frac{1}{p(n)},$$

where S makes at most $t(R_{A,k,m}, n, p)$ queries and $R_{A,k,m}$ is the worst-case running time of A on $E(k, m)$, taken over the coin tosses of A and E.

The definition of obfuscation can also be strengthened in a similar way. Obviously, the smaller t is, the stronger the security guarantee. For instance, if an encryption scheme (respectively, obfuscation) is t-secure then it (respectively, the obfuscator) can not do certain "stupid" things such as outputting the plaintext (respectively, the original function) in the clear on more than $\frac{nt(|E(.,.)|,n,n)}{n-1}$ keys (respectively, $\frac{nt(|O(.)|,n,n)}{n-1}$ functions). We note that the construction in Section 3 satisfies this definition for some specific t. However, the question remains as to how small t can be made.

Acknowledgements

We are grateful to Joan Feigenbaum and Hoeteck Wee for useful comments and discussions. In particular, permuting the obfuscated programs for obfuscating set-membership predicates and functions was gratefully pointed out by Hoeteck Wee. Also, we thank the anonymous referees for constructive feedback and suggestions that helped in improving the final draft. Notably, the weakness in the definition of obfuscation and digital lockers was highlighted by a referee.

References

1. Firefox password manager, http://www.firefoxtutor.com/61/securing-firefox-passwords/
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
3. Bosley, C., Dodis, Y.: Does privacy require true randomness? In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (2007)
4. Canetti, R.: Towards realizing random oracles:hash functions that hide all partial information. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997)
5. Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions. In: Proceedings of the 30th ACM Symposium on Theory of Computing, pp. 131–140 (1998)
6. Dodis, Y., Spencer, J.: On the (non)universality of the one-time pad. In: 43rd Symposium on Foundations of Computer Science (2002)
7. Futoransky, A., Kargieman, E., Sarraute, C., Waissbein, A.: Foundations and applications for secure triggers. eprint, 284 (2005)
8. Goldreich, O., Levin, L.: Hard-core predicates for any one-way function. In: Proceedings of the 21st ACM symposium on Theory of computing (1989)
9. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Science 28, 270–299 (1984)
10. Hofheinz, D., Malone-Lee, J., Stam, M.: Obfuscation for cryptographic purposes. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 214–232. Springer, Heidelberg (2007)
11. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004)
12. McInnes, J.L., Pinkas, B.: On the impossibility of private key cryptography with weakly random keys. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 421–435. Springer, Heidelberg (1991)
13. Wee, H.: On obfuscating point functions. In: Proceedings of the 37th ACM symposium on Theory of computing, pp. 523–532 (2005)

A Perfectly One-Way Probabilistic Hash Functions

A perfectly one-way hash function, POW for short, is a probabilistic function that satisfies collision resistance and hides all information about its input. Due to its probabilistic nature, such a function is coupled with an efficient verification algorithm that determines, given (x, y), whether y is a valid hash of x. Usually, collision resistance of *deterministic* hash functions requires that it is hard to find two input strings mapped to the same hash. However, because these functions are probabilistic by nature, we need to modify collision resistance to take the verification process into account. In particular, collision resistance says that it is hard to find two input and one output strings such that the verification scheme accepts the output as a valid hash of both input points. Formally,

Definition 6 (Public Verification, [4]). *A family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, satisfies* ***public verification*** *if there exists a deterministic polynomial-time algorithm V such that:*

1. *Completeness:* $\forall k \in K_n,\ x \in \{0,1\}^n,\ r \in R_n,\ \ V(x, H_k(x,r)) = 1.$
2. *Collision Resistance: For any nonuniform PPT, A:*

$$Pr[k \leftarrow K_n,\ (x_1, x_2, y) \leftarrow A(k) :\ x_1 \neq x_2 \wedge V(x_1, y) = V(x_2, y) = 1] < \mu(n).$$

There are several ways to formulate information hiding, some of which are not equivalent. We start with the most basic definition, namely semantic perfect one-wayness, and later present two more definitions, namely, statistical and computational indistinguishability. Semantic perfect one-wayness has its roots in semantic security of probabilistic encryption [9] which requires that every function that can be computed given the ciphertext can also be computed without it. However, the notion of secrecy in this setting is slightly weaker than semantic security because a hash can be used to verify whether a guess is correct or not. This notion is captured by a simulation-based definition which requires that every predicate computable given a hash can also be computed by a simulator with oracle access to the corresponding point function. Formally,

Definition 7 (Semantic Perfect One-wayness, [4]). *A family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, is called* ***semantically perfectly one-way*** *if it satisfies public verification (Definition 6) and, for any nonuniform PPT, A, and polynomial, p, there exists a nonuniform PPT S such that for sufficiently large n, any k, and any x:*

$$|Pr[r \leftarrow R_n,\ b \leftarrow A(k, H_k(x,r)) : b = 1]\ -$$

$$Pr[r \leftarrow R_n,\ b \leftarrow S^{F_x}(k) : b = 1]| \leq \frac{1}{p(n)}.$$

Recall F_x is the point function on x.

Remark 1. Note that semantic perfect one-wayness corresponds in a straightforward way to the virtual blackbox property required for obfuscating point functions in Definition 1. Thus, a function satisfying definition 7 is an obfuscation of a point function (with computational approximate functionality). However, the converse may not be true.

In more detail, let $\mathbb{H}$ be a semantic POW function. To obfuscate F_x, sample a seed, k, and random string, r, for $\mathbb{H}$ and output the obfuscation, $O(F_x) = k, H_k(x, r)$. The new function, $O(F_x)$, simply computes the predicate $V(., H_k(x, r))$. It can be shown that O is an obfuscator for the class of point functions. Completeness and collision resistance on $\mathbb{H}$ imply computational approximate functionality while semantic perfect one-wayness implies the virtual-blackbox property. On the other hand, an obfuscation of point functions may not be a POW function because approximate functionality does not rule out collisions chosen in an adversarial way.

As mentioned in the introduction, neither Definition 1 nor Definition 7 is sufficient for the security of our construction in Section 3 because they do not guarantee composition. Thus, we analyze our construction based on primitives with different composable properties. Two of these primitives are statistical and computational POW functions, which are defined in the rest of this appendix.

Statistical Perfect One-wayness. Statistical information hiding is captured by requiring statistical closeness between hashes of the same input and those of different inputs.

Definition 8 (Statistical t-Indistinguishability). *A family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called **statistically t-indistinguishable** if it satisfies public verification (Definition 6) and for any well-spread distribution $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ and any $k \in K_n$,*

$$\Delta(\underbrace{H_k(X_n, R_n^1), ..., H_k(X_n, R_n^{t(n)})}_{t(n)}, \underbrace{H_k(U_n^1, R_n^1), ..., H_k(U_n^{t(n)}, R_n^{t(n)})}_{t(n)}) \le \mu(n),$$

where each distribution R_n^i (respectively, U_n^i) is the same as R_n (respectively, U_n).

Moreover, if $\mathbb{H}$ is statistically t-indistinguishable for any polynomial, t, then it is called statistically indistinguishable.

We note that the first construction in [5] is slightly weaker than Definition 8 in that the input distribution has n^ϵ min-entropy instead of superlogarithmic min-entropy. Constructing functions with the latter property remains an open problem.

Computational Perfect One-wayness. Computational perfect one-wayness differs from statistical perfect one-wayness in two main ways. The first and obvious difference is that indistinguishability holds for *polynomially-bounded adversaries* only. Second, computational perfect one-wayness differs depending on whether we take the presence of auxiliary information into account. In this context, we restrict the notion of auxiliary information to uninvertible functions about the input.

Instead of explicitly writing two definitions, one with auxiliary information and another without it, we present here one definition only. To take both cases into account, we use the convention that auxiliary information is surrounded by boxes. So, by removing the words in boxes from Definition 9, we get the first definition while keeping the boxes gives us the second one. Formally,

Definition 9 (t-Indistinguishability)
Let $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ be any well-spread distribution. Let F be any (possibly probabilistic) uninvertible function. A family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where H_k :

$\{0,1\}^n \times R_n \rightarrow \{0,1\}^{l(n)}$ *for some polynomial* l*, is called* ***t-indistinguishable*** *with respect to* $\mathbb{X}$*,* $\boxed{\text{with auxiliary input } F}$*, if it satisfies public verification (Definition 6) and for any* $k \in K_n$ *and any PPT* A*:*

$$|Pr[x \leftarrow X_n,\ \boxed{z \leftarrow F(x)},\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(x, r_1), ..., H_k(x, r_t)) = 1] \quad -$$

$$Pr[x \leftarrow X_n,\ (u_1, ..., u_t) \leftarrow (U_n, ..., U_n),\ \boxed{z \leftarrow F(x)},\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(u_1, r_1), ..., H_k(u_t, r_t)) = 1]| \leq \mu(n).$$

If $\mathbb{H}$ *is t-indistinguishable* $\boxed{\text{with any auxiliary input } F}$ *with respect to any well-spread distribution* $\mathbb{X}$*, then it is called t-indistinguishable* $\boxed{\text{with auxiliary input}}$*. Moreover, if it is t-indistinguishable* $\boxed{\text{with auxiliary input}}$ *for any polynomial* t*, then it is called indistinguishable* $\boxed{\text{with auxiliary input}}$*.*

Isolated Proofs of Knowledge and Isolated Zero Knowledge

Ivan Damgård[1], Jesper Buus Nielsen[1], and Daniel Wichs[2]

[1] University of Aarhus, Denmark
[2] New York University, USA

Abstract. We consider proof of knowledge protocols where the cheating prover may communicate with some external adversarial environment during the run of the proof. Without additional setup assumptions, no witness hiding protocol can securely ensure that the prover knows a witness in this scenario. This is because the prover may just be forwarding messages between the environment and the verifier while the environment performs all the necessary computation.

In this paper we consider an ℓ-*isolated* prover, which is restricted to exchanging at most ℓ bits of information with its environment. We introduce a new notion called ℓ-isolated proofs of knowledge (ℓ-IPoK). These protocols securely ensure that an ℓ-isolated prover knows the witness. To prevent the above-mentioned attack, an ℓ-IPoK protocol has to have communication complexity greater than ℓ. We show that for any relation in NP and any value ℓ, there is an ℓ-IPoK protocol for that relation. In addition, the communication complexity of such a protocol only needs to be larger than ℓ by a constant multiplicative factor.

1 Introduction

A proof of knowledge [GMR85, BG92], is a protocol where a prover demonstrates to a verifier that he has a certain piece of information — typically the witness for some instance of an NP relation. Soundness of such a proof is usually formalized by insisting that there is a way to extract the witness from any prover who successfully convinces the verifier. The definition implicitly assumes that the prover talks to no one else during the proof. Intuitively, this may seem necessary to ensure that it is the prover himself who knows the witness, and not someone else helping the prover.

Nevertheless, in this paper we will consider a cheating prover who is able to communicate with some external adversarial entity, called the environment. We will insist that knowledge soundness still means that a witness can be extracted *from the prover himself*. From a technical point of view this means that an extractor is allowed to rewind the prover, but not the environment.

When the cheating prover can communicate *arbitrarily* with the environment, this notion can only be achieved by trivial protocols where the prover essentially hands the witness to the verifier. The obvious reason is that the witness may be located in the environment and the cheating prover only acts as a channel

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 509–526, 2008.

between the environment and the verifier while the environment gives an honest proof. In such a case, the cheating prover learns nothing more than the honest verifier during the proof and hence extraction implies that the honest verifier always learns a witness from a single run of the protocol. This simple attack requires the prover and the environment to communicate an entire transcript of an honest proof. We study what happens when such an attack is prevented by limiting the communication between the prover and the environment to be shorter than the communication used in the protocol.

One can imagine many ways such a partial isolation could be achieved in practical scenarios. If the prover is in close proximity to the verifier, they can be expected to communicate orders of magnitude faster than the prover can communicate with its environment. If in very close proximity, the fixed speed of light alone can be used to isolate a prover. Alternatively, consider a prover implemented on a smart card: for example, a smart card performing an identification protocol. The card reader could try to shield the card completely (e.g., using a Faraday cage) but this requires significant resources. It might be much easier to only prevent *large* amounts of communication. For example, the card reader could limit the amount of communication by measuring the energy consumption of the card. A significant amount of communication takes up a noticeable amount of energy, typically orders of magnitudes larger than what the card needs for standard operation.

To facilitate a formal study of such settings, we propose a notion of ℓ-isolated proofs of knowledge (ℓ-IPoK), where the cheating prover is restricted to communicating only ℓ bits of information with the environment during the run of the proof. Note that the number of bits of information communicated does not necessarily correspond to physical bits. For example, if the prover and the environment share (very) well synchronized clocks, then a short signal can communicate many bits of information based on the time it is sent. Later, we will also see that some of our protocols only need to restrict the number of exchanged *messages* where each message may contain arbitrarily many bits of information.

In practice, the physical setting determines the level of isolation and hence the communication threshold ℓ. For any such threshold, we would like to construct an ℓ-IPoK protocol. We therefore consider the notion of a parametrized IPoK compiler, or just IPoK, that generates an ℓ-IPoK protocol for any value of ℓ polynomial in the security parameter κ. Letting C denote the communication complexity of the generated proof system, we call $O = C/\ell$ the *overhead*. We saw that any non-trivial ℓ-IPoK protocol must have $C > \ell$, so an overhead greater than 1 is necessary.

It turns out to be easy to construct an IPoK with overhead $O = \text{poly}(\kappa)$ and with $\mathcal{O}(\ell+\kappa)$ rounds of communication. This is done by repeating a standard Σ-protocol $\ell + \kappa$ times so that there are many iterations where the prover cannot consult the environment. While this seems straightforward, it is not entirely trivial to prove that it works. Next, we show that, using novel techniques, it is also possible to construct an IPoK protocol with a *constant* overhead. This IPoK compiler generates protocols in which the number of rounds grows with the communication threshold ℓ. We show that this is necessary for any black-box extractable construction. However, using the non-programmable random oracle model as a non-black-box technique, we construct a constant round IPoK with an overhead that gets arbitrarily close to 1. Applying the non-black-box techniques introduced by Barak, we can get a constant-round construction based on a standard assumption. This last compiler, however, does not have a constant overhead. Our IPoK compilers are all non-trivial in that they produce protocols that are Zero Knowledge (ZK) in the standard sense (when the verifier is fully isolated), or at least witness indistinguishable (WI).

We also propose a notion of ℓ-isolated zero-knowledge (ℓ-IZK), where we require that a simulator can simulate any cheating verifier V^* that communicates at most ℓ bits with its environment during the proof. Since 0-IZK is essentially equivalent to the standard notion of ZK, it is known that every relation in NP has a 0-IPoK, 0-IZK protocol. On the other hand, consider a cheating verifier that simply acts as a channel between the environment and the honest prover while the environment runs the honest verifier code to generate challenge messages. A simulator for this scenario must essentially run an accepting proof with the environment, which means that it must know a witness. Since the simulator is only given the instance and not the witness, this implies that ℓ-IZK proof of knowledge protocols with communication complexity $C \leq \ell$ only exist for trivial languages where the witness is easy to find. On the positive side, we show how to construct an ℓ-IZK, ℓ-IPoK protocol for any NP relation R and any pre-defined threshold ℓ polynomial in the security parameter κ.

We conclude the paper by mentioning some applications of ℓ-IPoK using the physical assumption that one can ℓ-isolate a prover for the duration of the proof phase. Firstly, we can use a witness indistinguishable (WI) ℓ-IPoK to prevent "man-in-the-middle" attacks on identification schemes. Secondly, in a followup paper [DNW07], we show how to implement arbitrary multiparty computation securely in the UC framework without relying on any trusted third parties if the players can be partially isolated during a short, initial proof phase. This improves on the work of [Katz07], which showed that arbitrary MPC is possible in the UC framework when parties are fully isolated by putting their functionality on a tamper-proof hardware token. In some sense, our follow-up work justifies our choice of considering partially isolated parties for proofs of knowledge only rather than studying arbitrary multiparty computation in general, since the latter follows from the former.

2 Σ-Protocols

An NP relation R is a set of pairs (x, w) where $(x, w) \overset{?}{\in} R$ can be checked in poly-time in the length of x. For such a relation we define the witnesses for an instance x as $W_R(x) = \{w | (x, w) \in R\}$ and the language $L(R) = \{x | W_R(x) \neq \emptyset\}$.

We use Σ-protocols throughout the paper. A Σ-protocol is given by four PPT ITMs $(P, V, \mathcal{S}, \mathcal{X})$. In a Σ-protocol for relation R, the prover P is given $(x, w) \in R$ and the verifier V is given x. The protocol has three rounds: the prover $P(x, w)$ sends the first message a, the verifier $V(x)$ sends a uniformly random challenge $e \in \{0,1\}^l$, and P returns a response z. At the conclusion of the protocol, $V(x)$ outputs a judgment $J = \texttt{accept}$ or $J = \texttt{reject}$ based only on the conversation (x, a, e, z). An accepting conversation (x, a, e, z) is one for which V outputs accept. A Σ-protocol is called complete for R if $P(x, w)$ and $V(x)$ always produce accepting conversations. It is called special knowledge sound for R if, given two accepting conversations (x, a, e, z) and (x, a, e', z') with $e \neq e'$, the extractor $\mathcal{X}$ outputs $w = \mathcal{X}(x, a, e, z, e', z')$ such that $(x, w) \in R$. It is called special honest verifier zero-knowledge for R if for all $(x, w) \in R$ the simulator $\mathcal{S}$ on input (x, e) produces a simulated conversation (x, a, e, z) which is computationally indistinguishable from a conversation produced by $P(x, w)$ on challenge e. It is called statistical special honest verifier zero-knowledge for R if the distribution of simulated conversations is statistically close to the distribution of conversations produced by (P, V). A Σ-protocol is called a (statistical) Σ-protocol for R if it is complete, special knowledge sound and (statistical) special honest verifier zero-knowledge for R.

Many relations in cryptography have statistical Σ-protocols, but not all NP relations are known to have statistical Σ-protocols. If, however, there exists perfectly binding, computationally hiding commitment schemes then all NP relations have a Σ-protocol with computational special honest verifier zero knowledge.

Given two NP relations R_1 and R_2 one can define $R = R_1 \vee R_2$ by $((x_1, x_2), w) \in R$ iff $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$. Given two Σ-protocols Σ_1 and Σ_2 for R_1 respectively R_2 one can use the OR-construction [CDS94] to construct a Σ-protocol $\Sigma = \Sigma_1 \vee \Sigma_2$ for $R_1 \vee R_2$. This Σ-protocol will in addition be witness indistinguishable (WI) in the sense that a proof with instance x using witness w_1 is (at least computationally) indistinguishable from a proof with instance x using witness w_2 for an arbitrary (PPT) cheating verifier V^* — even if V^* is given w_1 and w_2. This in turn implies that the proof is witness hiding (WH) if the relations are hard: A cheating verifier which can compute a witness for R with non-negligible probability p, after seeing a proof, by definition computes a witness for either R_1 or R_2 with probability p. If we let P use a random witness, $w_l \in \{w_1, w_2\}$, then because of WI, the cheating verifier will compute the witness w_{3-l} not used by P with a probability negligibly close to $p/2$. This would contradict the hardness of R_{3-l}.

3 Isolated Proof of Knowledge and Isolated Zero-Knowledge

We start by introducing the notions of ∞-IPoK and ∞-IZK, and then discuss how to restrict the communication. An interactive proof system is defined by the PPT ITMs (P, V). We define the following notions:

Completeness. We let some PPT environment $\mathcal{Z}$ pick $(x, w) \in R$ and then run (P, V) on (x, w). We require that V accepts with all but negligible probability. For simplicity we consider only protocols running in some fixed number of rounds

ρ. The honest execution proceeds as in Fig. 1. We require that $\Pr[\textsc{Exec}^R_{P,V,\mathcal{Z}}(\kappa) = 0]$ is negligible in κ for all PPT $\mathcal{Z}$.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce $(x, w) \in R$. Then (x, w) is input to P and x is input to V.
execution: Then for $r = 1, \ldots, \rho$ the verifier V is activated to produce a message $v^{(r)}$ that is input to P which is activated to produce a message $p^{(r)}$ that is input to V. Then V is activated to produce a judgment $J \in \{\texttt{accept}, \texttt{reject}\}$. The output of the execution is a bit $\textsc{Exec}$, where $\textsc{Exec} = 1$ iff $J = \texttt{accept}$.

Fig. 1. Execution $\textsc{Exec}^R_{P,V,\mathcal{Z}}(\kappa)$ with honest parties

Knowledge Soundness. We model a cheating prover by replacing P with an arbitrary PPT ITM P^*. We assume that the cheating prover is able to communicate with its environment during the attack on V. In addition we now allow the environment to pick x which is not necessarily in $L(R)$. We augment the game with a PPT extractor $\mathcal{X}$ whose goal is to recover the witness w from the view of the prover (including its random coins and its communication with the verifier V and environment $\mathcal{Z}$) at the conclusion of any accepting run of the protocol. The extraction game is outlined in Fig. 2. We say that a protocol is an ∞-IPoK if for each PPT environment $\mathcal{Z}$ and each PPT cheating prover P^* there exists a PPT extractor $\mathcal{X}$ such that $\Pr[\textsc{Extr}^R_{P^*,V,\mathcal{Z},\mathcal{X}}(\kappa) = 0]$ is negligible in κ.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce x, and x is input to P^* and V.
execution: Then for $r = 1, \ldots, \rho$ the verifier V is activated to produce a message $v^{(r)}$ that is input to P^* which is activated to produce a message $p^{(r)}$ that is input to V. Besides this P^* can at any time output a message y to $\mathcal{Z}$ and get back a reply z. At the conclusion of the ρ rounds, the verifier V produces a judgment $J \in \{\texttt{accept}, \texttt{reject}\}$.
extraction: The output of an execution is a bit $\textsc{Extr}$. If $J = \texttt{reject}$ then $\textsc{Extr} = 1$. Otherwise we construct the view σ to be a concatenation of the random coins of P^*, the messages $v^{(r)}, p^{(r)}$ exchanged between prover and verifier, and all the messages exchanged between prover and environment. We let $w = \mathcal{X}(\kappa, \sigma)$. If $w \in W_R(x)$, then $\textsc{Extr} = 1$ and otherwise $\textsc{Extr} = 0$.

Fig. 2. Knowledge soundness extraction: $\textsc{Extr}^R_{P^*,V,\mathcal{Z},\mathcal{X}}(\kappa)$

If there exists one $\mathcal{X}$ which works for all provers P^* and all environments $\mathcal{Z}$, and $\mathcal{X}$ only uses rewinding black-box access to P^*, then we say that (P, V) is a black-box ∞-IPoK for R. Sometimes we allow a small cheat and let $\mathcal{X}$ run in *expected* polynomial time in which case we say that the protocol is an expected ∞-IPoK.

Zero Knowledge. We model a cheating verifier by replacing V with an arbitrary PPT ITM V^*. We assume that the cheating verifier is able to communicate with its environment during the attack on P. We model this by allowing V^* to communicate with $\mathcal{Z}$. We assume that the execution stops by $\mathcal{Z}$ outputting a bit. The execution with a cheating verifier is given in Fig. 3.

To define zero-knowledge we compare the execution $\text{Exec}^R_{P,V^*,\mathcal{Z}}$ to a simulation $\text{Sim}^R_{\mathcal{S},\mathcal{Z}}$, where $\mathcal{S}$ is an ITM acting as simulator. We want to capture that the proof does not leak any information on w to V^* which V^* could not have generated itself. We model the information that V^* can collect by what it is able to output to the environment. The job of the simulator $\mathcal{S}$ is then to demonstrate constructively that whatever V^* can leak to the environment could have been computed by V^* *without access to* P. The details are given in Fig. 4. Because simulation using a strict PPT simulator is hard, one usually allows a small cheat by letting $\mathcal{S}$ be *expected* PPT. We say that (P, V) is ∞-IZK for R if, for every PPT environment $\mathcal{Z}$ and every PPT cheating verifier V^*, there exists an expected PPT simulator $\mathcal{S}$ such that $|\Pr[\text{Sim}^R_{\mathcal{S},\mathcal{Z}}(\kappa) = 1] - \Pr[\text{Exec}^R_{P,V^*,\mathcal{Z}}(\kappa) = 1]|$ is negligible in κ.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce $(x, w) \in R$. Then (x, w) is input to P and x is input to V^*.
execution: Then for $r = 1, \ldots, \rho$ the cheating verifier V^* is activated to produce a message $v^{(r)}$ that is input to P which is activated to produce a message $p^{(r)}$ that is input to V^*. Besides this V^* can at any time output a message y to $\mathcal{Z}$ and get back a reply z. The execution stops by $\mathcal{Z}$ outputting a bit $\text{Exec} \in \{0, 1\}$.

Fig. 3. Execution $\text{Exec}^R_{P,V^*,\mathcal{Z}}(\kappa)$ with a cheating verifier

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce $(x, w) \in R$. Then x is input to $\mathcal{S}$.
execution: Then $\mathcal{S}$ can at any time output a message y to $\mathcal{Z}$ and get back a reply $z \in \{0, 1\}^*$. The execution stops by $\mathcal{Z}$ outputting a bit $\text{Sim} \in \{0, 1\}$.

Fig. 4. Simulation $\text{Sim}^R_{\mathcal{S},\mathcal{Z}}(\kappa)$

Isolation. The above definition of ∞-IZK, ∞-IPoK captures universally composable zero knowledge proofs of knowledge, as the cheating party is allowed arbitrary communication with its environment. We now describe how to model a corrupted party that is isolated from its environment. We start with the cheating prover in Fig. 2. We do not restrict how much P^* and $\mathcal{Z}$ communicate before or after the proof phase. However, from the point where P^* receives $v^{(1)}$ until it outputs $p^{(\rho)}$ we count the number of bits of information exchanged between $\mathcal{Z}$ and P^*. We say that P^* is $(\ell_\mathcal{Z}, \ell_P)$-isolated if P^* *sends* at most ℓ_P bits of information to $\mathcal{Z}$ and *receives* at most $\ell_\mathcal{Z}$ bits of information from $\mathcal{Z}$.

We restrict the cheating verifier in Fig. 3 in the same way, counting its communication with $\mathcal{Z}$ from sending $v^{(1)}$ until receiving $p^{(\rho)}$. We then say that (P, V) is an $(\ell_{\mathcal{Z}}, \ell_P)$-IPoK for R if, in the definition of knowledge soundness, we restrict ourselves to $(\ell_{\mathcal{Z}}, \ell_P)$-isolated cheating provers P^*. Similarly, we say that (P, V) is $(\ell_{\mathcal{Z}}, \ell_V)$-IZK for R if we restrict the definition of zero knowledge to only $(\ell_{\mathcal{Z}}, \ell_V)$-isolated cheating verifiers V^*. We define black-box and expected notions as above. We use ℓ-X to denote (ℓ, ℓ)-X.

Finally, we define the notion of a parametrized IPoK, or just IPoK, which takes the security parameter κ and the isolation parameter ℓ as inputs and produces an ℓ-IPoK protocol. An IPoK + IZK compiler produces a protocol which is ℓ-IPoK and ℓ-IZK. Letting $C(\kappa, \ell)$ denote the communication complexity of the produced ℓ-IPoK, we use $C(\kappa, \ell)/\ell$ to denote the overhead of the IPoK.

4 Constructing IPoK Compilers

4.1 A Simple Construction

Given any NP relation R, let Σ be a computational Σ-protocol for R. We present a simple construction of an IPoK compiler for R using the protocol Σ. For any ℓ and κ, let Σ^* be the proof system where Σ is run $\rho = \ell + \kappa$ times in sequence with one-bit challenges: For $r = 1, \ldots, \rho$, first P computes a first message a^r for Σ and sends it to V. Then the verifier sends a uniformly random $e^r \in \{0, 1\}$ and P returns the response z^r to V. The verifier V accepts iff (x, a^r, e^r, z^r) is accepting for all $r = 1, \ldots, \rho$.

Theorem 1. *The proof system Σ^* is an ℓ-IPoK for R. In addition, it is ZK in the standard sense of a fully isolated verifier.*

Proof. It is well known that there is an expected PPT simulator which simulates many repetitions of a Σ-protocol with 1 bit challenges for any isolated malicious verifier V^*. Hence Σ^* is 0-IZK. This also implies that it is witness indistinguishable (WI).

To see that Σ^* is ℓ-IPoK, let P^* be any cheating prover. The strong knowledge soundness extractor (recall Fig. 2) gets the transcript of a random accepting execution. Then, for each $r = 1, \ldots, \rho$, it rewinds P^* to the point just before e^r was sent to P^* and sends $e^{r\prime} = 1 - e^r$ instead. If P^* sends anything to $\mathcal{Z}$, then the extractor aborts the work on round r. Otherwise, it runs P^* (and replays any communication that was sent from $\mathcal{Z}$ to P^* in this stage during the actual proof) and gets a response $z^{r\prime}$. If $(x, a^r, e^{r\prime}, z^{r\prime})$ is accepting, then we can use the special knowledge soundness of Σ to compute $w \in W_R(x)$. Otherwise, the extractor proceeds to the next round. If no round yields $w \in W_R(x)$, then it gives up.

Clearly $\mathcal{X}$ is PPT. We want to show that the probability that P^* yields an accepting execution which $\mathcal{X}$ cannot extract is negligible; We call such an execution a winning execution since on such executions $\mathcal{Z}$ and P^* win the extraction game outlined in Fig. 2.

First let us frame the problem more abstractly. The random coins of P^* and $\mathcal{Z}$ together completely determine a strategy of how P^* responds to the challenges posed by V and the communication exchanged for each such message. We model such a strategy as a binary tree T. The edges of the tree represent the two possible challenges the verifier can send at any point in the protocol. The nodes of the tree represent the state of the prover P^* (and the environment $\mathcal{Z}$) at various stages in the protocol. An execution of the protocol between P^* and V corresponds to a random path from the root of the tree to a leaf.

We call a node e-correct if the prover that finds itself in the state represented by that node gives the correct response (one on which the verifier does not reject) for the challenge bit $e \in \{0, 1\}$, possibly after conferring with the environment. Otherwise we call the node e-incorrect. Similarly we call a node e-communicating if, on the challenge bit e, the prover sends some communication to the environment before giving a response.

Now let us look at the paths in the tree T that correspond to winning executions. For any node N along such a path, let e be the challenge bit that corresponds to the outgoing edge of N which lies on the path of the winning execution and let $\bar{e} = 1 - e$. Then

1. N is e-correct. This has to be the case since the path is accepting.
2. N is $\bar{e}$-incorrect or is $\bar{e}$-communicating. This has to be the case since otherwise the extractor would be able to extract a witness from this execution.

Now assume that two winning paths diverge from a node N. Then by property 1, N is 0-correct and 1-correct. By property 2, it then follows that N is 0-communicating and 1-communicating. But there can be at most ℓ such nodes on any path since the prover can communicate at most ℓ times. This shows that the (non-regular) subtree of T containing only winning paths contains at most 2^ℓ paths. There are $2^{\kappa+\ell}$ total paths in T and hence the probability of choosing a winning path is upper bounded by $1/2^\kappa$. We note that the above bound holds for any tree T and hence the probability of a bad execution occurring in a tree randomly chosen using the coins of P^* and $\mathcal{Z}$ is also upper bounded by $1/2^\kappa$ which is negligible in κ.

We note that in the above proof, we only need to limit communication *from* the environment *to* the prover. In other words, we actually described an (ℓ, ∞)-IPoK. In addition, the proof also works if we only restrict the *number of messages* from the environment to the prover but allow each message to contain arbitrary many bits of information.

4.2 A Constant Overhead Construction

As before, let R be a relation in NP and let Σ be a Σ-protocol for R with conversations (a, e, z). We use Σ as a building block from which we compile our ℓ-IPoK protocol. We again use many repetitions of a Σ-protocol with 1 bit challenges. However, the prover does not respond with the full value of z in each round, but only with a small share of z in some ramp secret sharing scheme. This way, the number of bits exchanged in each round is small. At the same time, if there are enough rounds in which the prover cannot communicate with the environment, the extractor can use rewinding and learn enough of the shares to recover the alternative response z' and hence the witness w. The protocol uses a perfectly binding, computationally hiding commitment scheme which can commit to m bits using a $\mathcal{O}(m)$-bit string. It also uses a family of secret-sharing schemes SSS over some finite field $\mathrm{GF}(2^v)$. We write a secret sharing of a message z as $(Z[1], \ldots, Z[N]) = \mathrm{SSS}(z; r)$, where r is the randomness used. Here, $Z[i]$ are the shares and they are elements in the field $\mathrm{GF}(2^v)$. We call $(Z[1], \ldots, Z[N])$ a codeword.

The values of M and N depend on the security parameter κ and the communication threshold ℓ:

- The input to the prover is $(x, w) \in R$, and the verifier gets x.
- The following interaction is repeated for $m = 1, \ldots, M$:
  1. We first have a *commit phase*. The prover computes:
     (a) A random first message a_m for Σ.
     (b) A response $z_m^{(b)}$ to first message a_m and challenge b for $b \in \{0, 1\}$.
     (c) A secret sharing $Z_m^{(b)} = \mathrm{SSS}(z_m^{(b)}; r_m^{(b)})$ of the secret $z_m^{(b)}$ using randomness $r_m^{(b)}$.
     (d) A commitment $c_m^{(b)}$ to the pair $(z_m^{(b)}, r_m^{(b)})$.
     The prover sends $(a_m, c_m^{(0)}, c_m^{(1)})$ to V.
  2. We now have a *read phase* of N rounds, where in each round $n = 1, \ldots, N$ the verifier may read the n'th field element in one of the codewords $Z_m^{(0)}$ or $Z_m^{(1)}$. Formally, for $n = 1, \ldots, N$
     (a) V chooses a challenge $e \in \{0, 1, \perp\}$ with probability distribution $\Pr(0) = \Pr(1) = \alpha/2$, $\Pr(\perp) = (1 - \alpha)$ and sends the challenge to V.
     (b) If $e \neq \perp$, P sends the field element $Z_m^{(e)}[n]$ to V. Else it sends back $\perp$.
     If the verifier tries to read more than αN field elements in a single codeword $Z_m^{(e)}$, then the prover stops the protocol, and the verifier rejects.
  3. Lastly, there is a *verification phase*, where the verifier is allowed to see the opening to one of $c_m^{(0)}$ or $c_m^{(1)}$ to check that during the read phase it got valid shares of a valid response:
     (a) V sends a uniformly random challenge $b \in \{0, 1\}$ to P.
     (b) P sends an opening of $c_m^{(b)}$ to V which then recovers $(z_m^{(b)}, r_m^{(b)})$.
     (c) V verifies that
        i. The shares of $z_m^{(b)}$ received during the read stage were calculated correctly from the sharing $Z_m^{(b)} = \mathrm{SSS}(z_m^{(b)}; r_m^{(b)})$.
        ii. The conversation $(x, a_m, b, z_m^{(b)})$ is an accepting conversation of Σ.

Fig. 5. The Constant-Overhead Protocol

We assume that there exists a constant $\alpha > 0$ such that for any N there is an instantiation of the secret sharing scheme which shares a message consisting of αN field elements and has a privacy threshold αN (any αN shares of the codeword reveal no information about the shared secret). In addition, the sharing allows efficient reconstruction when any αN of the shares $Z[i]$ are lost (i.e. replaced by $\perp$). We call α the rate of the secret-sharing scheme. Using this notation, the full protocol is described in Fig. 5.

Secret-sharing schemes with constant rate α, can be constructed using what is typically called "ramp schemes". A well-known ramp scheme can be constructed by modifying Shamir secret sharing so that the shares are defined by evaluating a polynomial of degree $2\alpha N - 1$ in which the secret makes up the top αN high degree coefficients and the remaining coefficients are random. This scheme has a rate of $\alpha \approx 1/3$ but requires us to use a field with $v \geq \log_2(N)$ which (as we will see later) will not give us a constant-overhead scheme. It is also possible to use a ramp scheme over a small (constant sized) finite field. Such schemes were studied recently in [CC06], [CCGHV07]. In particular, the result of [CCGHV07] shows how to use algebraic geometric codes to get a scheme with rate $\alpha = \frac{5}{21}$ in the field $\mathrm{GF}(2^v)$ with $v = 6$. The code is based on the curves of García and Stichtenoth [GS96] for which there are efficient constructions.[1]

Let $f(\kappa)$ be the communication complexity of the original Σ-protocol. In our construction we then pick

$$N \approx \max(\alpha^{-1} f(\kappa)\ ,\ 4^{-1}\alpha^{-1}\ell/\kappa)\ , \tag{1}$$

$$M \approx (\beta_L + \beta_F + 1)\kappa + 1\ , \tag{2}$$

where we define the constants

$$\beta_L \approx 16\alpha^{-1}\log_2(e)\ , \quad \beta_F \approx -1/\log_2(\alpha/4)\ . \tag{3}$$

The scheme SSS allows us to share a message consisting of $\alpha N \geq f(\kappa)$ field elements, each of length v bits, which gives the capacity of at least $f(\kappa)$ bits and hence enough to share a response z of the protocol Σ.

Communication Complexity. The communication complexity of all the commit phases and all of the verification phases is $\mathcal{O}(Mf(\kappa))$ The communication complexity of a single read phase is simply $(v + 2)N$ since it takes 2 bits to encode the challenge e and v bits to encode the response. The communication complexity of all the read phases is then $MN(v+2)$. Since $N \geq f(\kappa)$, the total communication complexity of the protocol is then $\mathcal{O}(MNv)$. Under the assumptions that $\ell \geq 4\kappa f(\kappa)$, equation (1) just becomes $N \geq 4^{-1}\alpha^{-1}\ell/\kappa$. Assuming, in addition, that v is constant, the communication complexity of the protocol simply becomes $\mathcal{O}(\ell)$ which means that the protocol has a constant overhead for large enough ℓ.

The round complexity of the protocol is $\mathcal{O}(MN)$ which, under the above assumptions on ℓ and v, is also $\mathcal{O}(\ell)$.

[1] Unfortunately, such codes do not exist for all N. However, for any N there is an N' in the interval $N \leq N' \leq 8N$ for which we can construct such a code. We ignore this subtlety in further discussion since it means at most a small constant blowup of our parameters.

Completeness. It is clear that an honest prover and an honest verifier generate an accepting conversation as long as the verifier does not try to read more than αN positions in the same codeword. The expected number of field elements an honest verifier reads in a particular codeword is $(\alpha/2)N$. Using the Chernoff bound, it is easy to see that the probability of reading more than αN elements in a single codeword is negligible in N and hence also in κ. Using union bound, we see that the probability of this happening for any one of the possible $2M$ codewords is still negligible in κ.

Knowledge Soundness Extractor. The extractor tries to reconstruct as much as possible of the codewords $Z_m^{(0)}, Z_m^{(1)}$ for each $m = 1, 2, \ldots, M$. It rewinds to each round in which the prover did not communicate during the original execution of the protocol. The extractor then tries both of the challenges $0, 1$ and replays any communication from the environment to the prover that occurred in the original execution. If the prover tries to send out a message to the environment, the extractor gives up on recovering that share of the codeword and replaces it with a loss symbol $\perp$. Otherwise the extractor successfully recovers the same share that the verifier would have gotten during the real execution. At the end of this process, the extractor will hold some candidate codewords $\tilde{Z}_m^{(0)}, \tilde{Z}_m^{(1)}$ for each epoch $m \in 1, \ldots, M$. For each such m, either the extractor can correctly decode both codewords and recover the witness or:

1. One of the codewords has more than αN loss symbols.
2. One of the codewords contains at least one share which is faulty because it does not correspond to the committed secret sharing.
3. One of the shared secrets $z_m^{(b)}$ is an incorrect response to the first message a_m and the challenge bit b.

We show that the probability of an accepting conversation having one of the above possibilities occur in *each* epoch $m = 1, \ldots, M$ is negligible in the security parameter κ. Intuitively, there cannot be too many states in which the prover will communicate resulting in a loss symbols, since the prover is limited to sending at most ℓ bits to the environment. There also cannot be too many times where the prover sends faulty shares in an accepting conversation because such conversations have a high likelihood of the prover being "caught" in the verification phase. For the same reason, there cannot be too many times where the prover shares an incorrect response z_m. We formalize this intuition in the full version of this paper and show that the probability of the prover being involved in an accepting conversation with the verifier on which the extractor subsequently fails to extract a witness, is upper bounded by $3 \times 2^{-\kappa}$. The proof only relies on restricting the number of bits *from* the prover *to* the environment. Also, as in the previous protocol, we only need to restrict the number of exchanged messages but can allow each message to contain arbitrary many bits of information.

The ZK Simulator. We now show that the protocol is also ZK in the standard sense, which also implies that it is WI. Here we simply modify the usual simulator for simulating many repetitions of a Σ-protocol with 1-bit challenges. On each epoch m, the simulator uses the special HVZK property to produce a random conversation (a, e, z) for Σ where e is a random bit. It then, in addition, produces a random secret sharing $\mathrm{SSS}(z; r)$ and a commitment $c_m^{(e)}$ to (z, r). In addition it produces a commitment $c_m^{(1-e)}$ to some garbage value. The simulator sends $(a, c_m^{(0)}, c_m^{(1)})$ to V^*. Then the simulator simulates the read phase of the protocol by responding with random field elements for the challenges $1 - e$ and with the secret shares of the codeword $\mathrm{SSS}(z; r)$ for challenges e. Lastly, in the verification phase, if the verifier V^* picks the challenge e then the simulator honestly opens $c_m^{(e)}$ and goes on to the next round. On the other hand, if V^* sends the challenge $1 - e$ then the simulator rewinds V^* to the beginning of the epoch and tries again. This is an expected polynomial time simulation and is indistinguishable from a real execution by the hiding property of the commitment scheme and the privacy of the secret sharing scheme.

4.3 Impossibility of a Constant Round Black-Box Extractable IPoK

In both of the IPoK constructions that we saw so far, the number of rounds grows with the communication threshold ℓ. Clearly, this has to be the case if we are only restricting the number of messages exchanged rather than the number of bits of information since otherwise the simple attack mentioned in the introduction will work. However, we now show that this is a necessary characteristic of any black-box extractable IPoK compiler even when we restrict the number of bits of information. In particular, once ℓ is super-logarithmic ($\ell = \omega(\log(\kappa))$), then no protocol with $\mathcal{O}(1)$ rounds can be a witness hiding ℓ-IPoK.

Theorem 2. *Any black-box construction of a witness hiding (expected) IPoK compiler, parametrized by the communication threshold ℓ and the security parameter κ, with ρ rounds of communication must satisfy $\ell/\rho = \mathcal{O}(\log(\kappa))$.*

Proof. We start with any protocol that runs in ρ rounds and let $q = \lfloor \ell/\rho \rfloor$. Let $f : \{0,1\}^* \times \{0,1\}^m \to \{0,1\}^q$ be a pseudorandom function with keys of size m. The existence of pseudorandom functions follows from that of one way functions which are guaranteed to exist if witness hiding proofs of knowledge exist at all. We define a class of provers with (hardcoded) values $r, s \in \{0,1\}^m$ and $(x, w) \in \mathcal{R}$. For each such prover we have the corresponding environment with the (hardcoded) value r (which acts as a shared key between environment and prover) and the hardcoded instance x. A prover P^* and the corresponding environment $\mathcal{Z}$ are chosen randomly from this class. The prover P^* acts just like an honest prover but checks in with the environment to make sure it has not been rewound prior to each round. The interaction is outlined in Fig. 6.

The outlined interaction has P^* send q bits on every round and receive q bits on every round. Since $q\rho \leq \ell$, the cheating prover is indeed ℓ-isolated. Assume that there is an extractor $\mathcal{X}$ which recovers a witness. Since the proof is witness hiding, the extractor must be able to get some more output from P^*, other than just one run of the protocol. However, the only way to do so in a black-box

The prover P^* begins by setting view to be the empty string. For $i = 1, \ldots, \rho$:

1. The verifier sends $v^{(i)}$ to P^*.
2. P^* sets $\mathsf{view} \leftarrow \mathsf{view}||v^{(i)}$, computes $\sigma_s^{(i)} \leftarrow f(\mathsf{view}; s)$, and sends $\sigma_s^{(i)}$ to $\mathcal{Z}$.
3. $\mathcal{Z}$ sends $\sigma_r^{(i)} \leftarrow f((\sigma_s^{(i)}, i); r)$ to P^*.
4. P^* verifies $\sigma_r^{(i)} = f((\sigma_s^{(i)}, i); r)$. If not then P^* quits. Otherwise P^* computes the response $p^{(i)}$ and sends it to V.

In the above interaction, $\mathcal{Z}$ has a counter to keep track of the round i. After it reaches $i = \rho$ and sends out $\sigma_r^{(\rho)}$, it aborts and stops responding to any incoming messages.

Fig. 6. Interaction between P^* and $\mathcal{Z}$ during a proof with V

manner is to rewind P^* and get an additional response $p'^{(i)}$ for some round i. This is only possible if $\mathcal{X}$ finds a collision on $f(\cdot; s)$ or guesses the value of $f(\cdot; r)$ on some point, which can happen in expected polynomial time if and only if $q = \mathcal{O}(\log(\kappa))$.

4.4 Non Black-Box Techniques

In Theorem 2, we showed that non-black-box techniques are needed to construct a constant-round IPoK compiler. The idea of using non-black box techniques based on standard cryptographic assumptions was first studied by Barak in [Bar01]. In the full version of this paper, we show how to use Barak's techniques to construct a constant round IPoK + IZK compiler.[2] Since the extractor for such a protocol does not rely on rewinding, it is also possible to construct protocols that are resettable-ZK [CGGM99, BGGL01]: that is the zero knowledge property holds even when the verifier can reset the prover and force it to run multiple times with the same random coins. This is especially pertinent to our setting where isolation might be achieved by putting a prover on a smart-card which can be easily reset. Barak's non-black-box techniques are, however, inefficient in practice (requiring an application of the Cook-Levin reduction) and it does not seem that they can be used to get a constant round protocol which also has a constant overhead.

Here we take a different approach and present a very efficient constant round protocol using random oracles. As before, let R be an NP-relation, and let Σ be

[2] For the reader familiar with Barak's basic idea, the adaption to the isolated setting is fairly straight forward: The basic idea is that the prover produces a commitment c to some machine M, then the verifier returns a long random string r, and the prover shows that either the claim holds or $M(c) = r$. The simulator takes $M = V^*$ to be the cheating verifier. In the isolated setting we prove that either the claim holds or $M(c, aux) = r$ for some auxiliary input aux of length at most ℓ bits. The simulator takes aux to be the communication between V^* and $\mathcal{Z}$. By letting r be longer than $\ell + \kappa$ bits the soundness is maintained.

1. First V sends a uniformly random string r of length $\kappa + \ell$ bits to P.
2. Then P starts running κ instances of Σ. It sends the first messages $a_1, ..., a_\kappa$ to V. Then, for $i = 1, \ldots, \kappa$:
   The prover P computes $z_i^{(0)}, z_i^{(1)}$, where z_i^e is the response to the first message a_i and the challenge bit e in Σ. The prover chooses random strings $r_i^{(0)}, r_i^{(1)}$ of length κ and sets $(s_i^{(0)}, s_i^{(1)}) = (H(r, r_i^{(0)}, z_i^{(0)}), H(r, r_i^{(1)}, z_i^{(1)}))$. Lastly, the prover sends $(s_i^{(0)}, s_i^{(1)})$ to V.
3. V sends random challenge bits $e_1, ..., e_\kappa$ to P.
4. For $i = 1, \ldots, \kappa$, P sends $z_i^{(e_i)}, r_i^{(e_i)}$ to V. By calling H, V checks that $s_i^{(e_i)} = H(r, r_i^{(e_i)}, z_i^{(e_i)})$, and also that $(a_i, e_i, z_i^{(e_i)})$ is an accepting conversation for Σ.

Fig. 7. A WI IPoK from a Random Oracle

a Σ-protocol for R. We assume an oracle H that takes inputs of size $3\kappa + \ell$ bits and outputs κ bits. The protocol is given in Fig. 7.

Theorem 3. *The proof system Σ^+ is ℓ-IPoK for R. In addition Σ^+ is WI if Σ is WI. The overhead of the given compiler is $1 + o(1)$ for large enough ℓ.*

Proof. The communication exchanged is that of κ runs of the Σ-protocol (which is poly(κ)) plus the randomness r and r_i and the tags s_i: a total of $\ell + \text{poly}(\kappa)$. This gives an overhead of $1+\text{poly}(\kappa)/\ell$ which is $1+o(1)$ for large enough ℓ. The protocol runs in 4 rounds.

The required extractor simply looks at all oracle calls made by P^* and tests if there exists two calls specifying inputs of form $(r, r_i^{(0)}, z_i^{(0)}), (r, r_i^{(1)}, z_i^{(1)})$ where the outputs were used by P^* to form a pair $(s_i^{(0)}, s_i^{(1)})$ and where V would accept both $z_i^{(0)}$ and $z_i^{(1)}$. If so, it computes the witness using the special soundness property of Σ, otherwise it gives up.

Since P^* can send at most ℓ bits to the environment, the environment has at least κ bits of uncertainty about r. Therefore all calls to H where r appears in the input must have been made by P^*, except with negligible probability. Furthermore, since oracle outputs are κ bits long, they cannot be guessed except with negligible probability. Hence, any value $s_i^{(e_i)}$ that is checked by V in Step 4 of the protocol, must have been generated by P^* calling H on an input $r, r_i^{(e_i)}, z_i^{(e_i)}$ that V would accept. We say that such an element $s_i^{(e_i)} = H(r, r_i^{(e_i)}, z_i^{(e_i)})$ generated by P^* calling H is well formed.

It follows that, except with negligible probability, the only way in which P^* can construct a set of pairs $\{(s_i^{(0)}, s_i^{(1)})\}$ that will make V accept and the extractor fail is if every pair $(s_i^{(0)}, s_i^{(1)})$ contains exactly 1 well formed element. But then V accepts with probability at most $2^{-\kappa}$.

If the underlying Σ-protocol is witness indistinguishable, then so are polynomially many repetitions of the protocol run in parallel. The only additional information the cheating verifier gets here are the hashes $s_i^{(\bar{e_i})} = H(r, r_i^{(\bar{e_i})}, z_i^{(\bar{e_i})})$ where $\bar{e_i} = 1 - e_i$ is the bit which the verifier did not pick as a challenge in

Step 3 of the protocol. However, these hashes look random (even if the verifier knows a witness w and can guess $z_i^{(\bar{e_i})}$) unless the verifier guesses $r_i^{(\bar{e_i})}$ which only happens with negligible probability. Hence the protocol is indeed WI.

We have stated the above result in the random oracle model for simplicity. In reality, we only use the oracle in a limited way. We do not need a "programmable" oracle, i.e., the technique where the security reduction gets to decide what the oracle should output. We rely on the random oracle model to ensure that an output cannot be computed in a distributed fashion between two parties, each having only some portion of the input (i.e., the cheating prover knowing r and the environment knowing $z_{i,e}$). We believe it should be possible to instantiate our oracle with a concrete function and a well defined non-black-box assumption (along the lines of the knowledge of exponent assumption) rather than basing ourselves on a heuristic.

4.5 From IPoK + WI to IPoK + IZK

For theoretical interest we include the following construction of an IPoK + IZK from a WI IPoK. In practice, this is only useful if we are in a situation where both the prover and the verifier can be assumed to be isolated. The construction is based on the FLS paradigm [FLS99].

Theorem 4. *Assuming the existence a perfectly binding, computationally hiding commitment scheme, there exists an IPoK + IZK compiler for every relation in NP.*

Proof. Let R be any NP relation. The verifier sends two commitments $C_0 = \text{commit}(m_0; r_0)$ and $C_1 = \text{commit}(m_1; r_1)$ to κ-bit random elements m_0 and m_1 using randomizers r_0 and r_1 respectively. Then V gives a WI ℓ-IPoK of (m, r) such that $C_1 = \text{commit}(m; r)$ or $C_2 = \text{commit}(m; r)$. It selects which witness (m_1, r_1) or (m_2, r_2) to use uniformly at random. If the proof is accepting, then P gives a WI ℓ-IPoK of (m, r, w) such that $C_0 = \text{commit}(m; r)$ or $C_1 = \text{commit}(m; r)$ or $(x, w) \in R$.

To show that the protocol is ℓ-IZK, the simulator runs V^* through the conclusion of the first WI IPoK protocol. If the proof given by V^* is accepting then, since V^* is ℓ-isolated, the simulator can extract some (m, r) such that $C_0 = \text{commit}(m; r)$ or $C_1 = \text{commit}(m; r)$. Then the simulator runs the second WI IPoK using the witness (m, r, ϵ), and the ℓ-IZK property follows from the witness indistinguishability of this proof.

To show that the protocol is ℓ-IPoK, the extractor simply extracts a witness in the WI proof given by the prover to get some (m, r, w) such that $C_0 = \text{commit}(m; r)$ or $C_1 = \text{commit}(m; r)$ or $(x, w) \in R$. If the extractor extracts a witness (m, r) for C_0 or C_1 then, with probability close to $\frac{1}{2}$, this differs from the witness used in the first ℓ-IPoK (by witness indistinguishability) and hence the prover and extractor together break the hiding property of the commitment scheme. Hence, with all but negligible probability, the extractor recovers a witness w such that $(x, w) \in R$.

5 Applications of WI IPoK

5.1 Preventing "Man-in-the-Middle" Attacks on Identification Schemes

An identification scheme is an interactive protocol where one party acts as a prover to securely prove its identity to another party acting as a verifier. Each prover has a public key which is known to all others. The usual solution has the prover perform a witness hiding proof of knowledge of the corresponding secret key. A "man-in-the-middle" attack on an identification scheme involves a cheating party simultaneously acting as a verifier for party A and a prover for party B. By simply redirecting messages between A and B the adversary is able to claim A's identity and successfully convince the party B. A previous solution for preventing such attacks, outlined in [CD97] requires a PKI in a strong sense: all the verifiers must have a registered public key for which they are guaranteed to know the secret key. Each prover then customizes his proof to a specific verifier so that the verifier is unable to redirect the proof to another party. Apart from requiring a strong PKI, in practice this also requires that the prover checks the identity of the verifier that he communicates with. For instance, if you use your mobile phone to do a proof of identity and get access to some resource R, the phone must display the identity of R, so you can verify that you actually meant to access R.

As an alternative solution, we propose using the physical assumption that the prover is ℓ-isolated from all parties aside from the verifier. In the introduction, we discussed some scenarios where this could be a reasonable assumption. The prover simply uses a witness hiding ℓ-IPoK to prove his identity. The ℓ-IPoK property ensures that the prover himself knows a witness even if he simultaneously acts as a verifier in another instance of the proof, while the witness hiding property ensures that the verifier cannot learn such a witness. This solution only requires that the verifier knows the correct public key for the prover, and for this a standard PKI suffices! In addition, the responsibility of not being fooled by man-in-the-middle attacks now falls, not on the prover, but on the verifier who must ensure that any prover he is interacting with is properly isolated. This places the burden on the physical design of the apparatus and so is much less prone to human mistakes.

5.2 Setting Up a PKI for General UC MPC

It is known that general multiparty computation secure in the UC framework is not possible without an honest majority and without any additional setup assumptions [CKL03]. To remedy this, previous work used setup assumptions such as the presence of a common reference string (CRS) or the existence of a public key infrastructure (PKI) where players are guaranteed to know the secret key corresponding to their registered public key. Both of the above assumptions require a trusted third party to initialize the setup. It is desirable to eliminate (or at least reduce) the level of trust required. We instead propose using the physical assumption that a player can be partially isolated during a portion of the computation. A variant of this setting was previously considered in [Katz07], which showed that one can implement arbitrary multiparty computation in the UC framework without any trusted third parties by using tamper proof hardware tokens. In particular, it is assumed that a player can isolate such a token so that it cannot communicate even a single bit of information with any other party. With ℓ-IPoK protocols, we can weaken the physical setup and only require that a party can be *partially* isolated from the environment during a portion of the computation. The parties once and for all register public keys with each other and provide proofs of knowledge of the corresponding secret keys using an ℓ-IPoK protocol where the prover functionality is ℓ-isolated from the environment. In a followup paper [DNW07], we show that this setup can be used as basis for UC secure multiparty computation tolerating an arbitrary number of adaptive corruptions. Note that, in particular, those results show that the witness indistinguishability property of the registration proof is sufficient and zero-knowledge is not required. This is an essential point, as in most settings it is unreasonable to assume that *both* of the interacting parties are isolated from the environment and we showed that one cannot achieve ZK without isolating the verifier to some extent.

6 Future Directions

The most interesting future research would be to improve the efficiency of the constructions we gave. In particular, it would be nice to have a smaller constant overhead than what we achieve in Section 4.2. Perhaps one could even find a black-box construction with an overhead of $1 + o(1)$ or show that such constructions are impossible. In addition, it would be interesting to come up with a specific reasonable non-black-box assumption (along the lines of the *knowledge of exponent* assumption) under which one could prove the security of the protocol in Fig. 7 or some similar protocol which runs in a constant number of rounds and has an overhead of $1 + o(1)$.

References

[Bar01] Barak, B.: How to go beyond the black-box simulation barrier. In: Proc. 42nd Annual Symposium on Foundations of Computer Science, Las Vegas, NV, USA, October 14–17, pp. 106–115. IEEE, Los Alamitos (2001)

[BG92] Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)

[BGGL01] Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-Sound Zero-Knowledge and its Applications. In: 42nd Annual Symposium on Foundations of Computer Science, Las Vegas, Nevada, October 14–17, IEEE, Los Alamitos (2001)

[CGGM99] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge Cryptology ePrint Archive 1999/022

[CD97] Cramer, R., Damgård, I.: Fast and Secure Immunization Against Adaptive Man-in-the-Middle Impersonations. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 75–87. Springer, Heidelberg (1997)

[CDS94] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)

[CKL03] Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)

[CC06] Chen, H., Cramer, R.: Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 521–536. Springer, Heidelberg (2006)

[CCGHV07] Chen, H., Cramer, R., Goldwasser, S., de Haan, R., Vaikuntanathan, V.: Secure Computation from Random Error Correcting Codes. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 291–310. Springer, Heidelberg (2007)

[DNW07] Damgård, I., Nielsen, J.B., Wichs, D.: Universally Composable Multiparty Computation with Partially Isolated Parties. Cryptology ePrint Archive 2007/332

[FLS99] Feige, U., Lapidot, D., Shamir, A.: Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions, Society for Industrial and Applied Mathematics. SIAM Journal on Computing 29(1), 1–28 (1999)

[GS96] García, A., Stichtenoth, H.: On the asymptotic behavior of some towers offunction fields over finite fields. J. Number Theory 61, 248–273 (1996)

[GMR85] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, Providence, Rhode Island, May 6–8, pp. 291–304 (1985)

[Katz07] Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)

David and Goliath Commitments: UC Computation for Asymmetric Parties Using Tamper-Proof Hardware

Tal Moran and Gil Segev

Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel
{tal.moran,gil.segev}@weizmann.ac.il

Abstract. Designing secure protocols in the Universal Composability (UC) framework confers many advantages. In particular, it allows the protocols to be securely used as building blocks in more complex protocols, and assists in understanding their security properties. Unfortunately, most existing models in which universally composable computation is possible (for useful functionalities) require a trusted setup stage. Recently, Katz [Eurocrypt '07] proposed an alternative to the trusted setup assumption: tamper-proof hardware. Instead of trusting a third party to correctly generate the setup information, each party can create its own hardware tokens, which it sends to the other parties. Each party is only required to trust that its own tokens are tamper-proof.

Katz designed a UC commitment protocol that requires both parties to generate hardware tokens. In addition, his protocol relies on a specific number-theoretic assumption. In this paper, we construct UC commitment protocols for "David" and "Goliath": we only require a single party (Goliath) to be capable of generating tokens. We construct a version of the protocol that is secure for computationally unbounded parties, and a more efficient version that makes computational assumptions only about David (we require only the existence of a one-way function). Our protocols are simple enough to be performed by hand on David's side.

These properties may allow such protocols to be used in situations which are inherently asymmetric in real-life, especially those involving individuals versus large organizations. Classic examples include voting protocols (voters versus "the government") and protocols involving private medical data (patients versus insurance-agencies or hospitals).

1 Introduction

Designing secure protocols that run in complex environments, such as those typically found in real-world applications, is a very challenging task. The design must take into account that such protocols may be executed concurrently with multiple other copies of the same protocol (e.g., many voters voting at the same time) or with different protocols (e.g., performing an electronic bank transaction in response to the results of an on-line auction). The *Universal Composability* (UC)

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 527–544, 2008.

framework was introduced by Canetti [5] to model the security of cryptographic protocols when executed in such environments. Protocols proven secure within the UC framework are, in particular, secure even under arbitrary composition.

Unfortunately, it turns out that unless a majority of the participating parties are honest (which can never be assumed when there are only two participants), almost no useful functionality can be securely realized in this framework [7,8]. On the positive side, however, Canetti and Fischlin [7] managed to circumvent these impossibility results by assuming a strong form of setup – a common reference string (CRS). This suffices for realizing any (well-formed) functionality in the UC framework while tolerating any number of dishonest parties [9].

The main drawback of assuming the availability of a CRS is that this requires trust in the party that constructs the CRS, and there are no security guarantees if the CRS is set in an adversarial manner. This state of affairs motivated the research of alternative setup assumptions that can circumvent the impossibility results, and imply the feasibility of securely realizing natural functionalities in the UC framework. A variety of alternative setup assumptions have already been explored, such as public-key registration services [1,6], signature cards [19] and a variation of the CRS assumption in which multiple strings are available [18].

The above mentioned setup assumptions still require trust in at least some parties in the system. Recently, Katz [20] proposed an alternative assumption that eliminates the need for such trusted parties. Katz suggested basing UC computations on a physical assumption: the existence of tamper-proof hardware. Under this assumption, an honest party can construct a hardware token T_F implementing any polynomial-time functionality F, but an adversary given the token T_F can do no more than observe the input/output characteristics of this token. Katz showed that such a primitive can be used, together with standard cryptographic assumptions, to realize the ideal multiple commitment functionality in the UC framework while tolerating any number of dishonest parties, and hence to realize general UC multi-party computation [9].

1.1 Our Contributions

We revisit Katz's proposal of basing universally composable computations on tamper-proof hardware. More specifically, we focus on realizing the ideal commitment functionalities (which suffice for realizing general UC multi-party computation [9]). In this work, we construct UC commitment protocols using tamper-proof hardware tokens that have several advantages over Katz's protocol:

David and Goliath: Asymmetric Assumptions Suffice. Katz's commitment protocol is symmetric with respect to the assumptions about the two parties: both parties must create a hardware token, and both must assume that their token is proof against tampering by the other party. In many situations, however, the two participating parties are not symmetric. For example, in a voting scenario, voters may not be able to create their own hardware, or may not trust that hardware they create (or buy) is truly "tamper-proof" against the government (the other party in a voting protocol).

Our commitment protocols only require a single party (Goliath) to generate a token. The other party (David) must ensure the token cannot communicate with Goliath, but does not have to make any assumptions about the power of Goliath. We construct different commitment protocols for Goliath as the sender and for Goliath as the receiver.

Reducing the Computational Assumptions. In addition to relying on the existence of tamper-proof hardware tokens, Katz's protocol relies on a specific number-theoretic assumption — the decisional Diffie-Hellman assumption. Katz posed as an open problem to rely on general computational assumptions (under the assumption that tamper-proof hardware exists). We answer this open problem and reduce the required computational assumptions. Our contributions in this regard are as follows:

- We demonstrate that computational assumptions are not necessary in order to realize the ideal functionality $\mathcal{F}_{\mathsf{COM}}$ (this functionality allows a single commitment for each hardware token) if we assume the existence of tamper-proof hardware tokens. That is, our protocols that realize $\mathcal{F}_{\mathsf{COM}}$ do not rely on any computational assumptions. These protocols are also secure against adaptive adversaries (although the proof for the adaptive case is deferred to the full version of the paper).
- We demonstrate that the existence of one-way functions suffices in order to realize the ideal functionality $\mathcal{F}_{\mathsf{MCOM}}$ (this functionality allows a polynomial number of concurrent commitments using the same hardware token) if we also assume the existence of tamper-proof hardware tokens.
- In keeping with the David and Goliath theme, even the protocols based on one-way functions do not make assumptions about Goliath's computational power.

"Bare-Handed" Protocols. The protocols presented in this paper are highly efficient, and in particular require David to perform only a few (elementary and simple) operations, such as comparing two bit-strings or computing the exclusive-or of two bit-strings (even in the protocol based on one-way functions). When David is the receiver, these operations can be performed by "bare-handed" humans without the aid of computers (when David is the sender, he may require a calculator). Such a property is useful in many situations where computers cannot be trusted or where "transparency" to humans is essential (which is the case, for example, when designing voting protocols). In addition, the same efficiency guarantees hold for the hardware tokens in the protocols that are secure against computationally unbounded adversaries; such tokens may be constructed using extremely constrained devices.

1.2 Related Work

Basing cryptographic protocols on physical assumptions. Basing cryptographic protocols on physical assumptions is a common practice. Perhaps the most striking example is the field of quantum cryptography, where the physics of quantum mechanics are used to implement cryptographic operations – some of which are impossible in the "bare" model. For example, Bennett and Brassard [2] achieved information-theoretically secure key agreement over public channels based only on assumptions about the physics of quantum mechanics.

Much work has been done on basing commitment schemes and oblivious transfer protocols on the physical properties of communication channels, using the random noise in a communication channel as the basis for security. Both commitment schemes and oblivious transfer protocols were shown to be realizable in the *Binary Symmetric Channel* model [13,14], in which random noise is added to the channel in both directions with some known probability. Later works show that they can also be implemented, under certain conditions, in the weaker (but more convincing) *Unfair Noisy Channel* model [15,16], where the error probability is not known exactly to the honest parties, and furthermore can be influenced by the adversary.

The work of Katz [20] was inspired by works of Chaum and Pedersen [11], Brands [4], and Cramer and Pedersen [12] that proposed the use of smartcards in the context of e-cash. The reader is referred to [20] for a brief description of their approach. More recently, Moran and Naor [21] demonstrated the possibility of implementing oblivious transfer, bit-commitment and coin flipping based on "tamper-evident seals" that model very intuitive physical models: sealed envelopes and locked boxes.

Concurrent Independent Work. Independent of this paper, Chandran, Goyal and Sahai [10] and Damgård, Nielsen and Wichs [17] address the problem of basing universally composable computations on tamper-proof hardware, and construct protocols realizing the ideal multiple commitment functionality $\mathcal{F}_{\mathsf{MCOM}}$.

Chandran et al. [10] show that $\mathcal{F}_{\mathsf{MCOM}}$ can be realized based on tamper-proof hardware tokens and one-way functions. The main advantage of their approach is that their security proof does not rely on the simulator's ability to rewind hardware tokens. This gives their protocol security against *reset* attacks (where the adversary can rewind the tokens). In particular, their protocol does not require hardware tokens to keep state between invocations. In addition, their protocol does away with the requirement that the parties know the code of the token which they distribute (we note that this assumption is essential both to Katz's construction and to ours).

Damgård et al. [17] focus on relaxing the "isolation" requirement and allow the hardware tokens to communicate with the outside world as long as the number of communicated bits in both direction is below some pre-determined threshold (which is polynomial in the security parameter). With this relaxation in mind, they realize $\mathcal{F}_{\mathsf{MCOM}}$ assuming the existence of one-way permutations and a semantically-secure dense public-key encryption scheme with pseudorandom ciphertexts.

The main advantage of our work over those of Chandran et al. [10] and Damgård et al. [17] is that in our constructions only one of the parties is required to create its own hardware token. As argued above, this is desirable and often essential in many scenarios. The constructions of Chandran et al. and Damgård et al. require that each party creates its own hardware token. That is, it is assumed that both parties have the resources required to create hardware tokens.

1.3 Paper Organization

The remainder of this paper is organized as follows. For those not familiar with the UC framework, we give some background in Section 2, as well as formal definitions of the different commitment functionalities. In Section 3, we briefly review the formal model for tamper-proof hardware tokens. In Section 4, we describe our protocols for UC commitment where Goliath is the sender, and sketch their proof of security. Section 5 does the same for the protocol in which David is the sender. Finally, Section 6 contains a discussion and some open problems.

2 The UC Commitment Functionalities

Many two-party functionalities can be easily implemented in a natural "secure" manner using a trusted third party that follows pre-agreed rules. In proving that a two-party protocol is secure, it is highly desirable to argue that the protocol behaves "as if it was performed using the trusted third party". The Universally Composability (UC) framework is a formalization of this idea. In the UC framework, the trusted third party is called the *ideal functionality*. The ideal functionality is described by an interactive Turing machine that can communicate by authenticated, private channels with the participants of the protocol. We refer the reader to [5] for a more detailed exposition.

In the UC framework two ideal commitment functionalities are considered: functionality $\mathcal{F}_{\mathsf{COM}}$ that handles a single commitment-decommitment process, and functionality $\mathcal{F}_{\mathsf{MCOM}}$ that handles multiple such processes. The advantage of $\mathcal{F}_{\mathsf{MCOM}}$ over $\mathcal{F}_{\mathsf{COM}}$ in our setting is that protocols that securely realize $\mathcal{F}_{\mathsf{MCOM}}$ may use the same hardware token for multiple commitments.

In this paper we consider an additional ideal commitment functionality, one that handles a bounded number of commitment-decommitment processes. We refer to this functionality as the ideal bounded commitment functionality $\mathcal{F}_{\mathsf{BCOM}}$. Formal descriptions of $\mathcal{F}_{\mathsf{COM}}$, $\mathcal{F}_{\mathsf{BCOM}}$ and $\mathcal{F}_{\mathsf{MCOM}}$ are provided in Figures 1, 2 and 3, respectively.

3 Modeling Tamper-Proof Hardware

Our formulation of tamper-proof hardware tokens is based on the one provided by Katz [20]. Katz defined an ideal "wrapper" functionality, $\mathcal{F}_{\mathsf{WRAP}}$, which captures the intuitive idea that an honest party can construct a hardware token T_F implementing any polynomial-time functionality F, but an adversary given the token T_F can do no more than observe the input/output characteristics of this

Functionality $\mathcal{F}_{\mathsf{COM}}$

1. Upon receiving $(\mathsf{commit}, sid, P, P', b)$ from P, where $b \in \{0,1\}$, record the value b and send $(\mathsf{receipt}, sid, P, P')$ to P' and the adversary. Ignore any subsequent commit messages.
2. Upon receiving $(\mathsf{open}, sid, P, P')$ from P, if some value b was previously recorded then send $(\mathsf{open}, sid, P, P', b)$ to P' and the adversary and halt. Otherwise halt.

Fig. 1. The ideal commitment functionality

Functionality $\mathcal{F}_{\mathsf{BCOM}}$

$\mathcal{F}_{\mathsf{BCOM}}$ is parameterized by a bound n on the number of allowed commitment-decommitment processes and an implicit security parameter k, and stores an internal counter j initialized to 0.

1. Upon receiving $(\mathsf{commit}, sid, cid, P, P', b)$ from P, where $b \in \{0,1\}$, if $j < n$ then set $j \leftarrow j+1$, record (cid, P, P', b) and send $(\mathsf{receipt}, sid, cid, P, P')$ to P' and the adversary. Ignore any subsequent $(\mathsf{commit}, sid, cid, P, P', \star)$ messages.
2. Upon receiving $(\mathsf{open}, sid, cid, P, P')$ from P, if some tuple (cid, P, P', b) was previously recorded then send $(\mathsf{open}, sid, cid, P, P', b)$ to P' and the adversary. Otherwise halt.

Fig. 2. The ideal bounded commitment functionality

Functionality $\mathcal{F}_{\mathsf{MCOM}}$

1. Upon receiving $(\mathsf{commit}, sid, cid, P, P', b)$ from P, where $b \in \{0,1\}$, record (cid, P, P', b) and send $(\mathsf{receipt}, sid, cid, P, P')$ to P' and the adversary. Ignore any subsequent $(\mathsf{commit}, sid, cid, P, P', \star)$ messages.
2. Upon receiving $(\mathsf{open}, sid, cid, P, P')$ from P, if some tuple (cid, P, P', b) was previously recorded then send $(\mathsf{open}, sid, cid, P, P', b)$ to P' and the adversary. Otherwise halt.

Fig. 3. The ideal multiple commitment functionality

token. An honest party, given a token $T'_{F'}$ by an adversary, has no guarantee regarding the function F' that this token implements (other than what the honest user can deduce from the input/output of this device).

Figure 4 describes our formulation of the ideal functionality $\mathcal{F}_{\mathsf{WRAP}}$. Informally, a party P is allowed to create a hardware token, which is then delivered to P'. We refer to P as the creator of the token, and to P' as the user of the token. The hardware token encapsulates a Turing machine M which is provided by the creator P. At this point, the functionality allows the user P' to interact with the Turing machine M in a black-box manner. That is, P' is allowed to send messages of its choice to M via the wrapper functionality, and receive the corresponding answers.

As in Katz's formulation, we assume that the tokens are partially isolated, in the sense that a token cannot communicate with its creator. Our formulation also allows the tokens to maintain state between invocations. Although tokens created by honest parties are only required to maintain a limited state (such as the current round number), we allow tokens created by adversarial parties to maintain arbitrary state across invocations.

Functionality $\mathcal{F}_{\mathsf{WRAP}}$

$\mathcal{F}_{\mathsf{WRAP}}$ is parametrized by a polynomial $p(\cdot)$ and an implicit security parameter k.

1. Upon receiving $(\mathsf{create}, sid, P, P', M)$ from P, where M is a description of a Turing machine, do:
   (a) Send $(\mathsf{create}, sid, P, P')$ to P'.
   (b) If no tuple of the form $(P, P', \star, \star)$ is stored, then store $(P, P', M, \perp)$.
2. Upon receiving $(\mathsf{run}, sid, P, msg)$ from P', do:
   (a) If no tuple of the form $(P, P', \star, \star)$ is stored, then halt. Otherwise, retrieve the unique stored tuple $(P, P', M, state)$.
   (b) Run $M(msg, state)$ for at most $p(k)$ steps, and denote the result by $(out, state')$. If M does not respond in the allotted time then set $out = \perp$ and $state' = state$.
   (c) Send (sid, P, out) to P', store $(P, P', M, state')$ and erase $(P, P', M, state)$.

Fig. 4. The ideal $\mathcal{F}_{\mathsf{WRAP}}$ functionality

4 Constructing Goliath Commitments

In this section we describe protocols that realize the ideal *bounded commitment functionality*, $\mathcal{F}_{\mathsf{BCOM}}$ (see Figure 2), and the ideal *multiple commitment functionality*, $\mathcal{F}_{\mathsf{MCOM}}$ (see Figure 3). In these protocols, only the sender creates a hardware token (i.e., the sender is the powerful Goliath). Our protocol for realizing $\mathcal{F}_{\mathsf{BCOM}}$ does not rely on any computational assumptions, and our protocol for realizing $\mathcal{F}_{\mathsf{MCOM}}$ relies on the existence of any one-way function. In specifying the protocols we treat the hardware token as one of the parties in the protocol. The code executed by the token (i.e., the description of the Turing machine M sent to $\mathcal{F}_{\mathsf{WRAP}}$) is implicitly described by the token's role in the protocol.

Notation. For a bit m, we denote $\boldsymbol{m} \stackrel{def}{=} m \circ \cdots \circ m \in \{0,1\}^k$ the k-bit string consisting of k copies of m. We denote the bitwise complement of a string m by $\bar{m}$. The bitwise *xor* of two strings, a and b is denoted $a \oplus b$, while the bitwise *and* of a and b is denoted $a \odot b$.

Protocol $\mathsf{BCOM}_{\mathsf{Goliath}}$

Joint input: a security parameter k, and a bound n on the number of allowed commitment-decommitment processes.

Setup phase. The sender chooses n random pairs $(a_i, b_i) \xleftarrow{R} \{0,1\}^k \times \{0,1\}^k$ and creates a token with these parameters. The token also contains state in the form of a counter j initialized to 0. The sender sends the token to the receiver.

Commit phase. Denote the sender's input by (cid, m), and denote by i the number of invocations of the commit phase so far between the sender and the receiver. We assume w.l.o.g. that $cid = i$ (otherwise, both sides can maintain a database that translates between the two).

1. The sender computes $x_i \leftarrow a_i \oplus b_i \oplus \boldsymbol{m}$, and sends (i, x_i) to the receiver.
2. The receiver chooses a random challenge $c_i \xleftarrow{R} \{0,1\}^k$ and sends (i, c_i) to the token.
3. The token verifies that its internal counter $j = i$ and $j < n$ (otherwise it sends $\perp$ to the receiver and halts). It increments the counter $j \leftarrow j + 1$. The token computes $y_i \leftarrow a_i \odot c_i \oplus b_i \odot \bar{c_i}$ and sends (i, y_i) to the receiver (this is equivalent to letting each bit of c_i choose whether to send the corresponding bit of a_i or the corresponding bit of b_i).

Opening phase.

1. The sender sends m and (i, a_i, b_i) to the receiver.
2. The receiver computes $z_i \leftarrow a_i \oplus b_i$ and verifies that $z_i \oplus x_i = \boldsymbol{m}$ and that $a_i \odot c_i \oplus b_i \odot \bar{c_i} = y_i$. If not, it outputs $\perp$ and halts. Otherwise, it outputs m.

Fig. 5. Protocol $\mathsf{BCOM}_{\mathsf{Goliath}}$

4.1 Realizing the Ideal $\mathcal{F}_{\mathsf{BCOM}}$ Functionality

The intuition underlying our commitment protocol is that the hardware token is already a form of commitment — the sender is committing to a program that is hidden by the wrapping functionality, and cannot be changed once it is sent. The sender "hides" the comitted value in the program. The problem is that the receiver should not be able to extract the value before the opening phase. We solve this by using the fact that the token cannot communicate with the sender; the receiver sends the token a random challenge whose response depends on the hidden value, but does not reveal it. Because the sender does not know the challenge, he will be caught with high probability if he attempts to equivocate in the opening phase.

A formal description of the protocol is provided in Figure 5, which is followed by a sketch of its security proof.

Security intuition. To see why the protocol is hiding, note that after the commit phase the value $a \oplus b$ remains uniformly distributed from the receiver's point of view, regardless of the value of m (since, for every index ℓ, the receiver can choose to learn either the ℓ^{th} bit of a or the ℓ^{th} bit of b, but not both).

The protocol is binding because in order to equivocate, the sender must change at least $\frac{1}{2}k$ bits of a and b in the opening phase. Because the sender does not know the challenge sent to the token, and hence does not know which bits of a and b the receiver has already seen, if it tries to equivocate it will be caught with overwhelming probability.

Proof sketch. For simplicity we sketch the proof of security only for the case of a static adversary, as this case already captures the important ideas in the proof. A complete proof for the case of an adaptive adversary will be provided in the full version of this paper. In order to prove that the protocol realizes $\mathcal{F}_{\mathsf{BCOM}}$ we need to construct a polynomial-time simulator $\mathcal{S}$ (an ideal adversary) such that for any polynomial-time environment machine $\mathcal{Z}$ and real-world $\mathcal{A}$, it holds that $\mathcal{Z}$ cannot distinguish between the ideal world and the real world with non-negligible advantage. In this sketch we focus on the two cases in which only one of the parties is corrupted. The cases in which both parties are corrupted or both parties are honest are dealt with in a straightforward manner.

The ideal-world adversary, $\mathcal{S}$, begins by setting up an internal simulation of all the real-world parties and functionalities: the sender, the receiver and $\mathcal{F}_{\mathsf{WRAP}}$ (this includes a simulation of the hardware token). Unless explicitly specified by the simulation protocols below, the simulated honest parties and $\mathcal{F}_{\mathsf{WRAP}}$ follow the honest protocol exactly. $\mathcal{S}$ keeps a "simulated view" for each honest party, consisting of the party's input (in the sender's case), its random coins, and the transcript of messages that party received throughout the simulation. At some points in the simulation, $\mathcal{S}$ may "rewrite" the simulated view of an honest party. It makes sure the new simulated view is consistent with any messages previously sent by that party to a corrupt party (note that $\mathcal{F}_{\mathsf{WRAP}}$ can never be corrupted, so messages sent to $\mathcal{F}_{\mathsf{WRAP}}$ may be changed as well).

Corrupted receiver. In the setup phase $\mathcal{S}$ simulates the interaction between $\mathcal{F}_{\mathsf{WRAP}}$, the honest sender and the corrupt receiver. That is, it chooses n random pairs (a_i, b_i) and sends to the simulated copy of $\mathcal{F}_{\mathsf{WRAP}}$ a description of the Turing machine which was specified by the protocol.

Whenever $\mathcal{S}$ receives a message $(\mathsf{receipt}, i)$ from $\mathcal{F}_{\mathsf{BCOM}}$, it chooses a random bit m'_i and simulates the honest sender with input (i, m'_i) interacting with the receiver and with the token. In this case it may be that $\mathcal{S}$ is simulating the honest sender with the wrong input, that is, in the ideal world the sender committed to m_i which is different from m'_i. We claim, however, that the view of the simulated receiver is independent of the committed bit and thus the view of $\mathcal{A}$ is identically distributed in both cases. The view of the simulated receiver consists of $x_i = a_i \oplus b_i \oplus \boldsymbol{m}'_i$ and $y_i \leftarrow a_i \odot c_i \oplus b_i \odot \bar{c}_i$. Each bit of y_i reveals either the corresponding bit of a_i or the corresponding bit of b_i. This implies that the value $a_i \oplus b_i$ is uniformly distributed from the receiver's point of view, and therefore x_i is uniformly distributed as well.

Whenever $\mathcal{S}$ receives a message (open, i, m_i) from $\mathcal{F}_{\mathsf{BCOM}}$ there are two possible cases. In the first case, it holds that m'_i (the bit with which $\mathcal{S}$ simulated the i-th commit stage) is the same as the revealed bit m_i. In this case $\mathcal{S}$ simulates the honest sender following the opening phase of the protocol. The simulation is clearly perfect in this case. In the second case, it holds that $m'_i \neq m_i$. If the simulated receiver sent some c_i to the token in the i-th commit phase, then $\mathcal{S}$ rewrites the history of the simulated sender by replacing a_i and b_i with $\hat{a}_i \leftarrow a_i \oplus \bar{c}_i$ and $\hat{b}_i \leftarrow b_i \oplus c_i$, and simulates the sender in the opening phase. Note that the new values satisfy $y_i = c_i \odot \hat{a}_i \oplus \bar{c}_i \odot \hat{b}_i$ and $\hat{a}_i \oplus \hat{b}_i \oplus x_i = \boldsymbol{m_i}$ (and are uniformly distributed given this view), and therefore the simulation matches the real world. If the receiver did not send c_i to the token in the i-th commit phase, then $\mathcal{S}$ internally rewinds the token to the point in which $\mathsf{commit}(i, m'_i)$ was invoked and reruns the simulation from that point, replacing m'_i with m_i and leaving all other inputs and random coins without change. Since the token is just part of the simulation of $\mathcal{F}_{\mathsf{WRAP}}$, and $\mathcal{S}$ knows the code it is executing, it can efficiently rewind it. Note that the messages seen by the receiver do not change and therefore the simulation is correct.

Corrupted sender. In the setup phase $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{WRAP}}$ to $\mathcal{A}$ who sends a description of a Turing machine M to $\mathcal{F}_{\mathsf{WRAP}}$. $\mathcal{S}$ now has a description of M and this will be used to rewind the simulated token at a later stage.

Whenever $\mathcal{A}$ initiates a new commit phase (say, the i-th commit phase), $\mathcal{S}$ simulates the honest receiver in this execution. If the commit phase was successful, then $\mathcal{S}$ needs to "extract" the bit m_i to which the corrupted sender committed to in order to instruct the sender in the ideal world to send this bit to $\mathcal{F}_{\mathsf{BCOM}}$. $\mathcal{S}$ rewinds the simulated token to the point in time before the honest receiver sent the challenge c_i to the token. $\mathcal{S}$ instead simulates sending $(i, \bar{c}_i)$ as the challenge. Denote the response y'_i. Now $\mathcal{S}$ "guesses" the values a_i and b_i using the original response y_i and the new response y'_i as follows: if the first bit of c_i was 1, then $\mathcal{S}$ sets the first bit of the guess for a_i to be the first bit of y_i, otherwise it sets the first bit of the guess for b_i to be the first bit of y'_i. Similarly $\mathcal{S}$ "guesses" all the bits of a_i and b_i. Denote by a'_i and b'_i these guesses. $\mathcal{S}$ sets m'_i as the majority of the bit-string $a'_i \oplus b'_i \oplus x_i$. $\mathcal{S}$ then instructs the ideal sender to send $(\mathsf{commit}, i, m'_i)$ to $\mathcal{F}_{\mathsf{BCOM}}$.

Whenever $\mathcal{A}$ initiates a new opening phase by sending (i, a_i, b_i) to the simulated receiver, $\mathcal{S}$ simulates the honest receiver in the opening phase. If $y_i = c_i \odot a_i \oplus \bar{c}_i \odot b_i$ and $x_i \oplus a_i \oplus b_i = \boldsymbol{m}_i$, where m_i is the bit that $\mathcal{S}$ instructed the ideal sender to send to $\mathcal{F}_{\mathsf{BCOM}}$ in the i-th commit phase, then $\mathcal{S}$ instructs the ideal sender to send (open, i) to $\mathcal{F}_{\mathsf{BCOM}}$. In this case we have that $m_i = m'_i$ and therefore the simulation is perfect. If the latter verification step fails, then $\mathcal{S}$ halts. The key point is that this happens only with negligible probability. That is, the probability that the corrupted sender manages to reveal its commitment to a bit different than m'_i is negligible.

In order to prove that the latter probability is indeed negligible, we consider the following game between two provers and a verifier: The verifier chooses a random bit c and sends it to the first prover. The first prover sends a bit y, and the second prover sends the bits (a, b, a', b'). The verifier accepts if $a \oplus b = \neg(a' \oplus b')$ and $y = a \odot c \oplus b \odot \bar{c} = a' \odot c \oplus b' \odot \bar{c}$. If the provers cannot communicate, the verifier will accept with probability at most 1/2 since for $a \oplus b = \neg(a' \oplus b')$

to hold, either $a \neq a'$ (in which case when $c = 1$ the verifier does not accept) or $b \neq b'$ (in which case when $c = 0$ the verifier does not accept).

In our protocol, if we consider only a single bit from each of the strings a_i, b_i, c_i, x_i and y_i, we can think of the sender and receiver as playing this game: the receiver is the verifier, and the sender and token are the two provers. The sender "wins" (causes the receiver to accept) if it opens that bit of the commitment to a different value than that "extracted" by the simulator in the commit phase. In the real protocol's commit phase, the game is played k times in parallel. Since the actual bit extracted by the simulator is the *majority* of the bits extracted from each of the games, in order for the sender to successfully open the commitment to a different bit, it must win in at least $k/2$ games. By the Parallel Repetition Theorem [22], the probability that the verifier accepts is exponentially small in the number of parallel repetitions.

4.2 Realizing the Ideal $\mathcal{F}_{\mathsf{MCOM}}$ Functionality

We show a simple variation of protocol $\mathsf{BCOM}_{\mathsf{Goliath}}$ that realizes the ideal $\mathcal{F}_{\mathsf{MCOM}}$ functionality. In the setup phase of $\mathsf{BCOM}_{\mathsf{Goliath}}$, the sender chooses several random pairs (a_i, b_i) and creates a token with these parameters. The number of commitments the protocol supports is therefore limited to the number of such pairs. However, if we are willing to rely on computational assumptions, the pairs (a_i, b_i) can be obtained as the output of a pseudorandom function on input i.

In the setup phase of the new protocol $\mathsf{MCOM}_{\mathsf{Goliath}}$ the sender chooses random seeds a and b for a family of pseudorandom functions $F = \{f_s\}$. The protocol then proceeds exactly as $\mathsf{BCOM}_{\mathsf{Goliath}}$ with the pairs $(f_a(i), f_b(i))$. A formal description of the protocol is provided in Figure 6.

In order to argue that protocol $\mathsf{MCOM}_{\mathsf{Goliath}}$ realizes $\mathcal{F}_{\mathsf{MCOM}}$ we first consider the protocol $\mathsf{BCOM}_{\mathsf{Goliath}}$ when parametrized with $n = 2^k$ (recall that k is the security parameter and n is the number of allowed commitments). With these parameters, the setup phase of the protocol consists of the sender choosing $n = 2^k$ random pairs (a_i, b_i), and the protocol allows the sender and the receiver to perform 2^k commitments – in particular it realizes $\mathcal{F}_{\mathsf{MCOM}}$ in the computational setting (ignoring the fact that the setup phase and the storage required by the token are exponential). Now, we claim that no polynomial-time adversary can distinguish between this protocol and the protocol $\mathsf{MCOM}_{\mathsf{Goliath}}$ with non-negligible probability, as any such adversary can be used in a straightforward manner to distinguish a random function chosen from the family F from a completely random function with non-negligible probability. Therefore, protocol $\mathsf{MCOM}_{\mathsf{Goliath}}$ realizes the ideal $\mathcal{F}_{\mathsf{MCOM}}$ functionality.

Since this proof relies on the seeds of the pseudorandom functions remaining secret from the adversary, it cannot be used to prove security against an adaptive adversary. In particular, the token's response in step 3 of the Commit phase may form a commitment to the seeds (a, b), in which case $\mathsf{MCOM}_{\mathsf{Goliath}}$ is *not* secure against an adaptive adversary.

Protocol $\mathsf{MCOM}_{\mathsf{Goliath}}$

Joint input: a security parameter k, and a family F of pseudorandom functions $f_s : \{0,1\}^k \rightarrow \{0,1\}^k$.

Setup phase. The sender chooses random seeds a and b for the family F and creates a token with these parameters. The token also contains state in the form of a counter j initialized to 0. The sender sends the token to the receiver.

Commit phase. Denote the sender's input by (cid, m), and denote by i the number of invocations of the commit phase so far between the sender and the receiver. We assume w.l.o.g. that $cid = i$ (otherwise, both sides can maintain a database that translates between the two).

1. The sender computes $x_i \leftarrow f_a(i) \oplus f_b(i) \oplus \boldsymbol{m}$, and sends (i, x_i) to the receiver.
2. The receiver chooses a random challenge $c_i \xleftarrow{R} \{0,1\}^k$ and sends (i, c_i) to the token.
3. The token verifies that its internal counter $j = i$ and $j < n$ (otherwise it sends $\perp$ to the receiver and halts). It increments the counter $j \leftarrow j + 1$. The token computes $y_i \leftarrow f_a(i) \odot c_i \oplus f_b(i) \odot \bar{c}_i$ and sends (i, y_i) to the receiver (this is equivalent to letting each bit of c_i choose whether to send the corresponding bit of $f_a(i)$ or the corresponding bit of $f_b(i)$).

Opening phase.

1. The sender sends m and $(i, f_a(i), f_b(i))$ to the receiver.
2. The receiver computes $z_i \leftarrow f_a(i) \oplus f_b(i)$ and verifies that $z_i \oplus x_i = \boldsymbol{m}$ and that $f_b(i) \odot c_i \oplus f_a(i) \odot \bar{c}_i = y_i$. If not, it outputs $\perp$ and halts. Otherwise, it outputs m.

Fig. 6. Protocol $\mathsf{BCOM}_{\mathsf{Goliath}}$

5 Constructing David Commitments

In this section we describe a protocol that realizes the ideal commitment functionality $\mathcal{F}_{\mathsf{COM}}$ (see Figure 1) without any computational assumptions, where only the receiver creates a hardware token (i.e., the sender is the limited David). In specifying the protocol we again treat the hardware token as one of the protocol participants. The code executed by the token (i.e., the description of the Turing machine M sent to $\mathcal{F}_{\mathsf{WRAP}}$) is implicitly described by the token's role in the protocol.

The intuition behind the protocol is that David can perform a commitment protocol *with the token*. Since there is no communication at all with Goliath, the commitment would be perfectly hiding. In such a case, however, David could postpone the interaction with the token to the opening phase (thus enabling him to equivocate). To overcome this problem, David must prove to Goliath *during the commit phase* that he has already interacted with the token. David does this by giving Goliath a "password" that was contained in the token. However, to prevent the token from using the password to give information about his commitment, David first "tests" Goliath to ensure that he already knows the

Protocol $\mathsf{COM}_{\mathsf{David}}$

Joint input: a security parameter k.

Setup phase. The receiver chooses four random values $s, t \xleftarrow{R} \{0,1\}^k$, $u, v \xleftarrow{R} \{0,1\}^{k/2}$, and creates a token with these parameters. The token contains a flag j initialized to 0. The receiver sends the token to the sender.

Commit phase. Denote the sender's input by $m \in \{0,1\}$.

1. The sender chooses two random values $a, b \xleftarrow{R} \{0,1\}^{k/2}$ and sends a to the token.
2. The token verifies that the internal flag $j = 0$ (otherwise it sends $\perp$ to the sender and halts). It sets $j \leftarrow 1$. The token computes $x \leftarrow u \cdot a + v$ (the computation is in $GF[2^{k/2}]$) and sends s, t and x to the sender.
3. The sender chooses a random challenge $c \xleftarrow{R} \{0,1\}^k$ and sends it the receiver, who replies with $y = s \cdot c + t$ (the computation is in $GF[2^k]$).
4. The sender verifies that $y = s \cdot c + t$, and in this case the sender sends to the receiver s, t, $z = \langle a, b\rangle \oplus m$ and b. If the verification fails then the sender halts.

Opening phase.

1. The sender sends m, a and x to the receiver.
2. The receiver verifies that $x = u \cdot a + v$ and that $z \oplus \langle a, b\rangle = m$. If not, it outputs $\perp$ and halts. Otherwise, it outputs m.

Fig. 7. Protocol $\mathsf{COM}_{\mathsf{David}}$

password. In the opening phase, David sends Goliath a second password (that does depend on the committed bit), which Goliath can verify.

A formal description of the protocol is provided in Figure 7, which is followed by a sketch of its security proof. A complete proof will be provided in the full version of the paper.

Proof sketch. We sketch the proof of security for the case of a static adversary. In order to prove that the protocol realizes $\mathcal{F}_{\mathsf{COM}}$ we need to construct simulator $\mathcal{S}$ (an ideal adversary) such that for any polynomial-time environment machine $\mathcal{Z}$ and real-world adversary $\mathcal{A}$, it holds that $\mathcal{Z}$ cannot distinguish between the ideal world and the real world with a non-negligible advantage. We note that our simulator in this proof runs in expected polynomial time (whereas the simulators in the previous section run in strict polynomial time). In this sketch we focus on the two cases in which only one of the parties is corrupted. The cases in which both parties are corrupted or both parties are honest are dealt with in a straightforward manner.

The ideal-world adversary, $\mathcal{S}$, begins by setting up an internal simulation of all the real-world parties and functionalities: the sender, the receiver and $\mathcal{F}_{\mathsf{WRAP}}$. Unless explicitly specified by the simulation protocols below, the simulated honest parties and $\mathcal{F}_{\mathsf{WRAP}}$ follow the honest protocol exactly. $\mathcal{S}$ keeps a "simulated view" for each honest party, consisting of the party's input (in the sender's case), its random coins, and the transcript of messages that party received throughout the simulation. At some points in the simulation, $\mathcal{S}$ may "rewrite" the simulated view of an honest party. It makes sure the new simulated view is consistent with any messages previously sent by that party to a corrupt party (note that $\mathcal{F}_{\mathsf{WRAP}}$ can never be corrupted, so messages sent to $\mathcal{F}_{\mathsf{WRAP}}$ may be changed as well).

Corrupted receiver. In the setup phase $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{WRAP}}$ to $\mathcal{A}$ who sends (on behalf of the simulated receiver) a description of a Turing machine M to $\mathcal{F}_{\mathsf{WRAP}}$. $\mathcal{S}$ now has a description of M and this will be used to rewind the simulated token at a later stage.

When $\mathcal{S}$ receives a message ($\mathsf{receipt}$) from $\mathcal{F}_{\mathsf{COM}}$, it chooses a random bit m' and simulates the honest sender with input m' interacting with the receiver and with the token. In this case it may be that $\mathcal{S}$ is simulating the honest sender with the wrong input, that is, in the ideal world the sender committed to m which is different from m'. We claim however, that the view of the simulated receiver is statistically-close to be independent of the committed bit.

The view of the simulated receiver consists of the challenge c, and of $s, t, z = \langle a, b\rangle \oplus m'$ and b. First, note that if the sender halts before sending the last message of the commit phase, the receiver's view is completely independent of the input bit (since it only affects the last message). So we only need to show that the view is statistically close to independent of m' conditioned on the sender completing the commitment phase successfully.

Since b is only sent in the last message, we can think of it being chosen then. Informally speaking, if there are many values of a for which the token returns some specific s and t, then by choosing b at random, with overwhelming probability $\langle a, b\rangle = 0$ for approximately half of them. Therefore $z = \langle a, b\rangle \oplus m'$ will be almost uniformly distributed, and hence almost independent of m'. If, on the other hand, there are only a few values of a for which the token returns some specific s and t, then the probability that the receiver given a random challenge c can predict $s \cdot c + t$ (and thus allow the sender to complete the commit phase) is negligible.

Whenever $\mathcal{S}$ receives a message (open, m) from $\mathcal{F}_{\mathsf{BCOM}}$ there are two possible cases. In the first case, it holds that m' (the bit with which $\mathcal{S}$ simulated the commit stage) is the same as the revealed bit m. In this case $\mathcal{S}$ simulates the honest sender following the opening phase of the protocol. The simulation is clearly perfect in this case. In the second case, it holds that $m' \neq m$. Denote by a the value that the simulated sender sent to the token in the simulated commit stage. The goal of the simulator is to rewind the simulated token and feed it with random values a' satisfying $\langle a', b\rangle \oplus z = m'$ until either 2^k iterations have passed or until the token outputs (s', t', x') where s' and t' are the same s and t that the token output when it was given a. If more than 2^k iterations have passed, then $\mathcal{S}$ fails and halts. Otherwise, $\mathcal{S}$ simulates the honest sender in the opening phase by sending a', x' and m'. Clearly, if $\mathcal{S}$ does not halt and manages to find such a', then the simulation is correct. In what follows we argue that the expected running time of the simulator is polynomial in the security parameter k.

We show that for any set of random coins of the corrupted receiver, the expected running time of $\mathcal{S}$ is upper bounded by a (fixed) polynomial. Fix the random coins of the receiver, then the receiver and the token define two functions: the token defines the function $T(a) = (s, t, x)$ and the receiver defines the function $y(c)$. Then the expected number of iterations performed by $\mathcal{S}$ is given by

$$\mathrm{E}\left[\mathsf{Iterations}\right] = \sum_{s,t,c} \Pr_a\left[s,t\right] \cdot \Pr_c\left[c\right] \cdot \mathrm{E}\left[\mathsf{Iterations}|s,t,c\right] \ .$$

Notice that if the tuple (s, t, c) is not consistent, in the sense that $s \cdot c + t \neq y(c)$, then the simulator halts. In addition, conditioned on s and t, the expected number of iterations is independent of c and is equal $1/\Pr_a\left[s,t\right]$ (here we ignore the requirement that $\langle a', b\rangle \oplus z = m'$, since b was chosen after s and t were determined, and therefore this requirement will only multiply the expected running time by some constant). Therefore,

$$\begin{aligned}\mathrm{E}\left[\mathsf{Iterations}\right] &= 2^{-k} \cdot \sum_{\substack{\text{consistent } (s,t,c) \\ \text{s.t. } \Pr[s,t]>0}} \Pr_a\left[s,t\right] \cdot \frac{1}{\Pr_a\left[s,t\right]} \\ &= 2^{-k} \cdot \sum_{c} \left|\left\{(s,t) : \Pr_a\left[s,t\right] > 0 \text{ and } (s,t,c) \text{ is consistent}\right\}\right| \ .\end{aligned}$$

We conclude the argument by showing that the above sum is at most $O(2^k)$ which implies that $\mathrm{E}\left[\mathsf{Iterations}\right]$ is constant. Consider the bipartite graph $G = (L, R, E)$ defined as follows. The left set L is the set of all pairs (s, t) for which $\Pr_a\left[s,t\right] > 0$. Notice that since $a \in \{0,1\}^{k/2}$ then $|L| \leq 2^{k/2}$. The right set R is the set of all possible c values, i.e., the set $\{0,1\}^k$. Finally, an edge $((s,t),c)$ exists if the tuple (s, t, c) is consistent, i.e., satisfies $s \cdot c + t = y(c)$. The above sum is exactly the number of edges in the graph: for every $c \in R$ we count the number of incoming edges $((s,t),c)$. The useful property of this graph is that it does not contain any cycles of length 4: it is straightforward to verify that there cannot be two different left-side vertices (s_1, t_1) and (s_2, t_2), and two different right-side vertices c_1 and c_2 that form a cycle of length 4). We can thus use the following theorem to conclude that the number of edges in the graph is at most $O(2^k)$:

Theorem 1 ([3], Chapter 6, Theorem 2.2). *Let $Z(m, n; s, t)$ be the minimal number such that any bipartite graph with vertex parts of orders m and n and $Z(m, n; s, t)$ edges must contain as a subgraph $K_{s,t}$ (the complete bipartite graph with vertex parts of orders s and t). Then*

$$Z(m, n; s, t) < (s-1)^{1/t}(n-t+1)m^{1-1/t} + (t-1)m \ .$$

Note that $K_{2,2}$ is a cycle of length 4, hence the number of edges in the graph is bounded by $Z(2^k, 2^{k/2}; 2, 2) < (2^{k/2}-1)(2^k)^{1/2} + 2^k < 2^{k+1}$.

Corrupted sender. In the setup phase $\mathcal{S}$ simulates the interaction between $\mathcal{F}_{\mathsf{WRAP}}$, the corrupted sender and the honest receiver. That is, it chooses (on behalf of the receiver) random values s, t, u and v, and sends to the simulated copy of $\mathcal{F}_{\mathsf{WRAP}}$ a description of the Turing machine which was specified by the protocol.

Whenever $\mathcal{A}$ initiates a commit phase, $\mathcal{S}$ simulates the honest receiver in this execution. If the commit phase was successful, then $\mathcal{S}$ needs to "extract" the bit m to which the corrupted sender is committed in order to instruct the sender in the ideal world to send this bit to $\mathcal{F}_{\mathsf{COM}}$. If the corrupted sender sent a value a to the simulated token, then $\mathcal{S}$ computes $m' = \langle a, b \rangle \oplus z$ and instructs the sender in the ideal world to send (commit, m') to $\mathcal{F}_{\mathsf{COM}}$. if the sender did not send any value to the token, $\mathcal{S}$ chooses a random bit m' and instructs the sender in the ideal world to send (commit, m') to $\mathcal{F}_{\mathsf{COM}}$.

Whenever $\mathcal{A}$ initiates a new opening phase by sending (a, x, m) to the simulated receiver, then $\mathcal{S}$ simulates the honest receiver in the opening phase. If $x = u \cdot a + v$ and $z \oplus \langle a, b \rangle = m$, where u, v, z and b are the values from the commit phase, and $m = m'$ where m' is the bit that the ideal sender sent to $\mathcal{F}_{\mathsf{COM}}$ in the commit phase, then $\mathcal{S}$ instructs the ideal sender to send (open) to $\mathcal{F}_{\mathsf{COM}}$. In this case the simulation is perfect. If the latter verification step fails, then $\mathcal{S}$ halts. The key point is that this happens only with negligible probability. That is, the probability that the corrupted sender manages to reveal its commitment to a bit different than m' is negligible. This is because in order to open his commitment to a different bit, the sender must send some $a' \neq a$ in the opening phase. However, to successfully pass verification, the sender must guess the value for $u \cdot a' + v$ (having seen, at most, $u \cdot a + v$). Since these values are independent, the sender guesses correctly with negligible probability.

6 Discussion and Open Problems

Multiple commitment functionality for David. Our protocol for commitment when David is the sender only realizes the $\mathcal{F}_{\mathsf{COM}}$ functionality. Unfortunately, this is an inherent limitation in the protocol rather than an artifact of the proof; when commit is invoked multiple times using the same hardware token, the token's messages when opening a commitment can reveal information about other commitments (that have not yet been opened). Constructing an $\mathcal{F}_{\mathsf{MCOM}}$ functionality for David is an interesting open problem.

We note that the protocol can be composed serially using a single token (as long as every commitment is opened before the next one is invoked). Using the same idea as we did in Goliath's commitment protocol, we can then replace the random values with a pseudorandom function to extend the functionality to any polynomial number of serial invocations (this means that the actual number of hardware tokens needed depends only on the maximum number of concurrent commitments).

Human-compatible commitment for David. David's commitment protocol (cf. Figure 7) requires David to perform a multiplication and an addition operation in a large finite field. While this may be possible to do on paper (or with a calculator), it would be useful to find a protocol that can be performed using simpler operations (as is the case with Goliath's commitment protocols).

Strict polynomial-time simulation for David's commitment. The simulator for David's commitment protocol, unlike Goliath's, runs in expected polynomial time when Goliath is corrupted (rather than strict polynomial time). Although we prove this protocol is secure even for a computationally unbounded Goliath, it is still an interesting question whether a protocol can be constructed with a strict polynomial-time simulator.

Relaxing the physical assumption to tamper-*evident* hardware. Katz's protocol can be implemented using tamper-evident, rather than tamper-proof hardware, if the tokens are returned to their creators after the setup phase. Our bounded commitment protocol can also use this relaxed assumption; since the queries to the token do not depend on the bit to be committed, they can be made ahead of time and the token returned to its owner. This method will not work if the number of commitments is not known ahead of time. Finding a David/Goliath protocol for $\mathcal{F}_{\mathsf{MCOM}}$ based on tamper-evident rather than tamper-proof hardware is an interesting problem.

References

1. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: Proceedings of the 45th Annual Symposium on Foundations of Computer Science, pp. 186–195 (2004)
2. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. In: Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, pp. 175–179 (1984)
3. Bollobas, B.: Extremal Graph Theory. Courier Dover Publications (1978)
4. Brands, S.: Untraceable off-line cash in wallets with observers. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 302–318. Springer, Heidelberg (1994)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pp. 136–145 (2001); An updated version is available from the Cryptology ePrint Archive, Report 2000/067
6. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Proceedings of the 4th Theory of Cryptography Conference, pp. 61–85 (2007)
7. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
8. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. Journal of Cryptology 19(2), 135–167 (2006)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: Proceedings of the 34th Annual ACM Symposium on the Theory of Computing, pp. 494–503 (2002)
10. Chandran, N., Goyal, V., Sahai, A.: Improved UC secure computation using tamper-proof hardware. Cryptology ePrint Archive, Report 2007/334 (2007)
11. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)

12. Cramer, R., Pedersen, T.P.: Improved privacy in wallets with observers. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 329–343. Springer, Heidelberg (1994)
13. Crépeau, C.: Efficient cryptographic protocols based on noisy channels. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 306–317. Springer, Heidelberg (1997)
14. Crépeau, C., Kilian, J.: Achieving oblivious transfer using weakened security assumptions. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, pp. 42–52 (1988)
15. Damgård, I., Fehr, S., Morozov, K., Salvail, L.: Unfair noisy channels and oblivious transfer. In: Proceedings of the 1st Theory of Cryptography Conference, pp. 355–373 (2004)
16. Damgård, I., Kilian, J., Salvail, L.: On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 56–73. Springer, Heidelberg (1999)
17. Damgård, I., Nielsen, J.B., Wichs, D.: Universally composable multiparty computation with partially isolated parties. Cryptology ePrint Archive, Report 2007/332 (2007)
18. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
19. Hofheinz, D., Müller-Quade, J., Unruh, D.: Universally composable zero-knowledge arguments and commitments from signature cards. In: Proceedings of the 5th Central European Conference on Cryptology (2005)
20. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
21. Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, pp. 285–297 (2005)
22. Raz, R.: A parallel repetition theorem. SIAM Journal on Computing 27(3), 763–803 (1998)

New Constructions for UC Secure Computation Using Tamper-Proof Hardware

Nishanth Chandran*, Vipul Goyal**, and Amit Sahai***

Department of Computer Science, UCLA
{nishanth, vipul, sahai}@cs.ucla.edu

Abstract. The Universal Composability framework was introduced by Canetti to study the security of protocols which are concurrently executed with other protocols in a network environment. Unfortunately it was shown that in the so called plain model, a large class of functionalities cannot be securely realized. These severe impossibility results motivated the study of other models involving some sort of setup assumptions, where general positive results can be obtained. Until recently, all the setup assumptions which were proposed required some trusted third party (or parties).

Katz recently proposed using a *physical setup* to avoid such trusted setup assumptions. In his model, the physical setup phase includes the parties exchanging tamper proof hardware tokens implementing some functionality. The tamper proof hardware is modeled so as to assume that the receiver of the token can do nothing more than observe its input/output characteristics. It is further assumed that the sender *knows* the program code of the hardware token which it distributed. Based on the DDH assumption, Katz gave general positive results for universally composable multi-party computation tolerating any number of dishonest parties making this model quite attractive.

In this paper, we present new constructions for UC secure computation using tamper proof hardware (in a stronger model). Our results represent an improvement over the results of Katz in several directions using substantially different techniques. Interestingly, our security proofs do not rely on being able to rewind the hardware tokens created by malicious parties. This means that we are able to relax the assumptions that the parties *know* the code of the hardware token which they distributed. This allows us to model real life attacks where, for example, a party may simply pass on the token obtained from one party to the other without actually knowing its functionality. Furthermore, our construction models the interaction with the tamper-resistant hardware as a simple request-reply protocol. Thus, we show that the hardware tokens used in our construction can be *resettable*. In fact, it suffices to use token

* Research supported in part by grants listed below.

** Research supported in part by grants listed below.

*** This research was supported in part by NSF ITR and Cybertrust programs (including grants 0627781, 0456717, 0716389, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Okawa Research Award, and an Alfred P. Sloan Foundation Research Fellowship.

N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 545–562, 2008.

which are completely stateless (and thus cannot execute a multi-round protocol). Our protocol is also based on general assumptions (namely enhanced trapdoor permutations).

1 Introduction

The universal composability (UC) framework of security, introduced by Canetti [Can01], provides a model for security when protocols are executed multiple times in a network where other protocols may also be simultaneously executed. Canetti showed that any polynomial time computable multi-party functionality can be realized in this setting when a strict majority of the players are honest. Canetti and Fischlin [CF01] then showed that without an honest majority of players, there exists functionalities that cannot be securely realized in this framework. Canetti, Kushilevitz and Lindell [CKL06] later characterized the two-party functionalities that cannot be securely realized in the UC model ruling out almost all non-trivial functions. These impossibility results are in a model without any setup assumptions (referred to as the "plain" model). These results can be bypassed if one assumes a setup in the network. Canetti and Fischilin suggest the use of common reference string (CRS) and this turns out to be a sufficient condition for UC-secure multi-party computation for any polynomial time functionality, for any number of dishonest parties [CLOS02]. Some other "setup" assumptions suggested have been trusted "public-key registration services" [BCNP04,CDPW07a], government issued signature cards [HMQU05] and so on.

UC Secure Computation based on Tamper Proof Hardware. Recently, Katz [Kat07] introduced the model of tamper resistant hardware as a setup assumption for universally composable multi-party computation. An important attraction of this model is that it eliminates the need to trust a party, and instead relies on a physical assumption. In this model, a party P creates a hardware token implementing a functionality and sends this token to party P'. Given this token, P' can do nothing more than observe the input/output characteristics of the functionality. Based on the DDH assumption, Katz gave general feasibility results for universally composable multi-party computation tolerating any number of dishonest parties.

Our Contributions. In this paper, we improve the results of Katz in several directions using completely different techniques. Our results can be summarized as follows:

– Knowing the Code: A central assumption made by Katz [Kat07] is that all parties (including the malicious ones) *know* the program code of the hardware token which they distributed. This assumption is precisely the source of extra power which the simulator gets in the security proofs [Kat07]. The simulator gets the power of *rewinding the hardware token* which is vital for the security proofs to go through. However we argue that this assumption might be undesirable in practice. For example, it does not capture real life adversaries who may simply pass on hardware tokens obtained from one party to another. As noted by Katz [Kat07], such attacks may potentially be prevented by making the creator of a token easily identifiable (e.g., the token could output the identity of the creator on certain fixed input). However, we note that a non-sophisticated fix of this type might to susceptible to attacks where a malicious party builds a *wrapper* around the received token to create a new token and passes it on to other parties. Such a wrapper would use the token inside it in a black-box way while trying to answer the user queries. Secondly, one can imagine more sophisticated attacks where tokens of one type received as part of one protocols may be used as tokens of some other type in other protocols. Thus, while it may be possible to design constructions based on this assumption, it seems like significant additional analysis might be needed to show that this assumption holds.

We relax this assumption in this work. In other words, we make no assumptions on how malicious parties create the hardware token which they distribute.

- Resettability of the Token: The security of the construction in [Kat07] also relies on the ability of the tamper-resistant hardware to maintain state (even when, for example, the power supply is cut off)[1]. In particular, the parties need to execute a two-round interactive protocol with the tamper-resistant hardware. It is explicitly assumed that the hardware cannot be *reset* [CGGM00]. In contrast, our construction models the interaction with the tamper-resistant hardware as a simple one round request-reply protocol. Thus, we are able to show that the hardware tokens used in our construction can be *resettable*. In fact, it suffices to use token which are completely stateless (and thus cannot even execute a multi-round protocol). We argue that relaxing this assumption about the capability of the tamper resistant tokens is desirable and may bring down their cost considerably.
- Cryptographic Assumptions: An open problem left in [Kat07] was to construct a protocol in this model which is based on *general assumptions*. We settle this problem by presenting a construction which is based on enhanced trapdoor permutations previously used in CLOS [CLOS02] and other works.

Our communication model for the token also has an interesting technical difference from the one in [Kat07]. In [Kat07], it is assumed that once P creates a hardware token and hands it over to P', then P cannot send any messages to the token (but can receives messages from it). We require the opposite assumption; once the token has been handed to P', it cannot send any messages to P (but can potentially receive messages from it). It is easy to see that if the communication

[1] As Katz [Kat07] noted, this assumption can be relaxed if the token has an inbuilt source of randomness and thus messages sent by the token in the protocol are different in different execution (even if the other party is sending the same messages). Note that a true randomness source is needed to relax this assumption and cryptographic techniques such as pseudo random functions do not suffice.

is allowed in both directions, then this model degenerates to the plain model which is the subject of severe impossibility results [CF01,CKL06].

Our Techniques. Recall that all the participating parties exchange tamper proof hardware tokens with each other before the protocols starts. To execute the protocol, the parties will presumably make queries to the tokens received from other parties. We observe that the simulator (in the proof of security) can be given *access to all the queries which any dishonest party makes to a token designed by an honest party.* Our first idea to exploit this extra power (and make the simulator non-rewinding) is to extract the inputs of the dishonest parties as follows. If party P_1 wants to commit to its input to party P_2, it will first have to feed the opening to the commitment to the token provided by P_2 which will output a signature on the commitment (certifying that it indeed saw the opening). One may observe that this is very close in spirit to how proofs are done in the Random Oracle model. One problem which we face is that P_1 cannot give a signature obtained from the token directly to P_2 (since these signatures can potentially help establish a covert communication channel between the token and P_2). Thus, the party P_1 instead gives a commitment to the signature obtained (and will later prove that this commitment is a commitment to valid signature).

While the above basic idea is simple and elegant, significant more work is required to turn it into a construction that achieves our main goals (in a way that the construction relies only on general assumptions). The first issue we face is: how to prove that the commitment given is a commitment of a valid signature? While executing a UC commitment scheme, P_1 might be interacting with multiple parties at the same time. We use concurrent zero-knowledge (ZK) proofs [DNS98,PRS02] for this purpose. Although concurrent ZK proofs are not directly usable as building blocks in larger protocols (since they are secure only under concurrent *self composition* rather than general composition), we show that they can be used as a building block in our case by presenting a direct analysis of the resulting scheme to prove its security (under concurrent attacks).

The most difficult issue which we face is: how to prove that a dishonest P_1 cannot commit to a valid signature without actually making a query to P_2's token. This is because if P_1 commits to a valid signature (and even gives a proof of knowledge of the commitment) without making a query to the token, the UC-simulator cannot rewind P_1 to extract this signature (and contradict security of the underlying signature scheme). We get around this issue by showing that the analysis of this case can be separated from the UC-Simulator. In a separate *extraction abort lemma* proven "outside the UC framework", we show that if this case happens, the Environment has the capability to forge signatures (in other words, we rewind the environment and extract a forged signature). Thus, we reduce the failure probability of our simulator to the probability with which the signatures can be forged. Similarly, we have a *decommitment abort lemma* proven outside the UC framework where we reduce the success probability of an adversary opening to a different string than the one committed to (in a UC commitment scheme we construct) to the soundness of an underlying (sequentially secure) zero knowledge proof. Other problems that we deal with are:

the issue of selective abort by the hardware token (where the token refuses to give a valid signature for some particular inputs only) and the issue of equivocating the commitment while keeping the UC-Simulator straightline.

We are able to incorporate all the above ideas into a construction that achieves the multiple commitment functionality in the UC framework. We remark that in the end, the analysis of our construction is admittedly somewhat complex. While one can consider alternative approaches to how a device would extract, several problems like the issue of selective abort (which was simpler to deal with in our approach) again seem to imply that the final solution (which would take care of all these problems) will be no simpler.

Concurrent Independent Work. Independent of our work, Damgard et al [DNW07] proposed a new construction for UC secure computation in the tamper proof hardware model. The main thrust of their work seems to obtain a scheme where the hardware tokens only need to be *partially isolated*. In other words, there exists a pre-defined threshold on the number of bits that the token can exchange with the outside world (potentially in both directions). Their construction is also based on general assumptions (albeit their assumptions are still stronger than ours).

Damgard et al [DNW07] however do not solve the main problems addressed by this work. In particular, their work is in the same *rewinding based simulator* paradigm as Katz [Kat07] and thus requires the same assumption that the sender is aware of the program code of the hardware token which it distributed. Furthermore, the security of their construction relies upon the assumption that the hardware token is able to keep state (i.e., is not resettable).

2 Our Model

Our model is a modification of the model in [Kat07]. The central modifications we need are to allow for adversaries who may supply hardware tokens to other parties without knowing the code of the functionality implemented by the hardware token. To model adversaries who give out tokens without actually "knowing" the code of the functionality of the tokens, we consider an ideal functionality $\mathcal{F}_{Adv}$ that models the adversarial procedure used to create these tokens. The security of our protocol will be defined over all probabilistic polynomial time (PPT) adversaries $\mathcal{F}_{Adv}$. The ideal functionality $\mathcal{F}_{wrap}$ implements the tamper-resistant hardware as in [Kat07].

We first formally define the $\mathcal{F}_{wrap}$ functionality which is a modification of the $\mathcal{F}_{wrap}$ functionality of [Kat07]. This functionality formalizes the intuition that an honest user can create a hardware token T_F implementing any polynomial time functionality, but an adversary given the token T_F can do no more than observe its input/output characteristics. $\mathcal{F}_{wrap}$ models the hardware token (sent by P_i to P_j) encapsulating a functionality M_{ij}. The only changes from [Kat07] we make is that M_{ij} is now an Oracle machine (instead of a 2-round interactive Turing machine) and does not require any externally supplied randomness.

$\mathcal{F}_{wrap}$ models the following sequence of events: (1) a party P_i (also known as *creator*) takes software implementing a particular functionality M_{ij} and seals this software into a tamper-resistant hardware token, (2) The creator then gives this token to another party P_j (also known as the *receiver*) who can use the hardware token as a black-box implementing M_{ij}. Figure 1 shows the formal description of $\mathcal{F}_{wrap}$ based on an algorithm M_{ij} (modified from [Kat07]). Note that M_{ij} could make black box calls to other tokens implementing M_{xy} (to model the tokens created by an adversarial party) in a way that the circularity problems are avoided.

$\mathcal{F}_{wrap}$ is parameterized by a polynomial p and an implicit security parameter k. There are 2 main procedures:

Creation. Upon receiving $(create, sid, P_i, P_j, M_{ij})$ from P_i or from $\mathcal{F}_{Adv}$, where P_j is another user in the system and M_{ij} is an Oracle machine, do:

1. Send $(create, sid, P_i, P_j)$ to P_j.
2. If there is no tuple of the form $(P_i, P_j, \star)$ stored, then store (P_i, P_j, M_{ij}).

Execution. Upon receiving (run, sid, P_i, msg) from P_j, find the unique stored tuple (P_i, P_j, M_{ij}) (if no such tuple exists, then do nothing). Run M_{ij} with input msg for at most $p(k)$ steps and let out be the response (set $out = \perp$ if M_{ij} does not respond in the allotted time). Send (sid, P_i, out) to P_j.

Fig. 1. The $\mathcal{F}_{wrap}$ functionality

We now formally describe the Ideal/Real world for multi-party computation in the tamper-proof hardware model. Let there be n parties $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ (P_i holding input x_i) who wish to compute a function $f(x_1, x_2, \cdots, x_n)$. Let the adversarial parties be denoted by $\mathcal{M} \subset \mathcal{P}$ and the honest parties be denoted by $\mathcal{H} = \mathcal{P} - \mathcal{M}$. We consider only static adversaries. As noted before, to model adversaries who give out tokens without actually "knowing" the code of the functionality of the tokens, we consider an ideal functionality $\mathcal{F}_{Adv}$ that models the adversarial procedure used to create these tokens. $\mathcal{F}$ is the ideal functionality that computes the function f that the parties $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ wish to compute, while $\mathcal{F}_{wrap}$ (as discussed earlier) models the tamper-resistant device.

Real World. Our real world is the $(\mathcal{F}_{Adv}, \mathcal{F}_{wrap})$-hybrid world. In the real world, when a party P_i begins a protocol with another party P_j it exchanges a hardware token with P_j. We note that this exchange of token need be done only once in the protocol. This is modeled as follows. If P_i is malicious, then P_i sends arbitrary messages to $\mathcal{F}_{Adv}$ functionality ($\mathcal{F}_{Adv}$ could use this information for the code creation of the adversarial token to be sent to P_j). At the end of this interaction, $\mathcal{F}_{Adv}$ sends a program code (corresponding to the token that is to be given to P_j) to $\mathcal{F}_{wrap}$. This program code can make black box calls to tokens of other (possibly honest) parties. If P_i is honest, then P_i sends a program code directly to $\mathcal{F}_{wrap}$ that will serve as the code for the hardware token to be sent to P_j. During protocol execution, all queries made to tamper-resistant hardware tokens are made to the $\mathcal{F}_{wrap}$ functionality. The parties execute the protocol and compute the function $f(x_1, x_2, \cdots, x_n)$.

Ideal World. The ideal world is the $(\mathcal{F}_{Adv}, \mathcal{F}_{wrap}, \mathcal{F})$-hybrid world. The simulator S simulates the view of the adversarial parties. As in the real world, when a party P_i begins a protocol with party P_j it has to specify the code for the hardware token to be sent to P_j. If P_i is adversarial, then P_i initially sends arbitrary messages to $\mathcal{F}_{Adv}$. $\mathcal{F}_{Adv}$ sends a program code (corresponding to the token that is to be given to P_j) to $\mathcal{F}_{wrap}$. This program code can make black box calls to tokens of other parties. If P_i is honest, then the simulator S generates the program code for the token to be sent to P_j (S does this honestly according to the protocol specifications for creating the program code). S sends this program code to $\mathcal{F}_{wrap}$. When an adversarial party queries a token created by another adversarial party, the simulator S forwards the query to $\mathcal{F}_{wrap}$ and then upon receiving the response from $\mathcal{F}_{wrap}$, forwards it to the querying party. When an adversarial party queries a token created by an honest party, the simulator S replies with the response to the querying party on its own. Honest parties send their inputs to the trusted functionality $\mathcal{F}$. Simulator extracts inputs from adversarial parties and sends them to $\mathcal{F}$. The ideal functionality $\mathcal{F}$ returns the output to all honest parties and to the simulator S who then uses it to complete the simulation for the malicious parties.

Remark. To be able to model an adversary which takes honest party tokens received in one protocols and uses them as subroutines for creating its tokens in some other protocol, we consider the GUC framework introduced by Canetti et al [CDPW07b]. The proofs in this paper can be modified so as to prove that our protocol for a functionality $\mathcal{F}$ $F_{wrap}-$EUC-realizes $\mathcal{F}$. This ensures that $\mathcal{F}_{wrap}$ has tokens created by honest parties even as part of other protocols.

3 Preliminaries

As in [Kat07], we will show how to securely realize the multiple commitment functionality $\mathcal{F}_{mcom}$ in the $(\mathcal{F}_{Adv}, \mathcal{F}_{wrap})-$ hybrid model for static adversaries. This will imply the feasibility of UC-secure multi-party computation for any well formed functionality ([CF01,CLOS02]). The primitives we need for the construction of the commitment functionality are non-interactive perfectly binding commitments, a secure signature scheme, pseudorandom function and concurrent zero knowledge proofs (that are all implied by one-way permutations [GL89,NY89,HILL99,Gol01,Gol04,DNS98,PRS02]).

Non-Interactive Perfectly Binding Bit Commitment. We denote the non-interactive perfectly binding commitment to a string or bit a (from [GL89]) by $\mathsf{Com}(a)$. $\mathsf{Open}(\mathsf{Com}(a))$ denotes the opening to the commitment $\mathsf{Com}(a)$ (which includes a as well as the randomness used to create $\mathsf{Com}(a)$).

Secure signature scheme. We use a secure signature scheme (security as defined in [GMR88]) with public key secret key pair (PK, SK) that can be constructed from one-way permutations ([NY89]). By $\sigma_{PK}(m)$ we denote a signature on message m under the public key PK. We denote the verification algorithm by $\mathsf{Verify}(PK, m, \sigma)$ that takes as input a public key PK, message m and purported signature σ on message m. It returns 1 if and only if σ is a valid signature of m under PK.

Concurrent Zero knowledge. Informally, concurrent zero knowledge proofs (introduced by [DNS98]) are zero-knowledge proofs that remain zero knowledge even when executed in the concurrent setting. In the concurrent setting, several protocols may be executed at the same time, with many verifiers talking simultaneously with one or more provers. Adversarial verifiers may interleave executions of different protocols and may base their messages on partial executions of other protocols. We shall use the concurrent zero knowledge protocol of Prabhakaran, Rosen and Sahai [PRS02]. For further details we refer the reader to [PRS02].

4 The Construction

We show how to securely realize the multiple commitment functionality $\mathcal{F}_{mcom}$ in the $(\mathcal{F}_{Adv}, \mathcal{F}_{wrap})-$ hybrid model for all PPT static adversaries and for all PPT $\mathcal{F}_{Adv}$. We will first give a construction that realizes the single commitment functionality in the $(\mathcal{F}_{Adv}, \mathcal{F}_{wrap})-$ hybrid model for static adversaries and then note that this can be extended to realize $\mathcal{F}_{mcom}$. P_1 wishes to commit to a string a (of length n bits) to P_2.

Token Exchange phase. P_2 generates a public-key/secret-key pair (PK, SK) for a secure signature scheme, a seed s for a pseudorandom function $F_s(\cdot)$ and sends a token to P_1 encapsulating the following functionality M_{21}:

– Wait for message $I = (\mathsf{Com}(b), \mathsf{Open}(\mathsf{Com}(b)))$. Check that the opening is a valid opening to the commitment. If so, generate signature $\sigma = \sigma_{PK}(\mathsf{Com}(b))$ and output the signature. The randomness used to create these signatures is obtained from $F_s(I)$.

We note that the token exchange phase can take place any time before P_2 begins a protocol with P_1 and needs to take place only once.

Commitment phase. We denote the protocol in which P_1 commits to a string a (of length n bits) to P_2 by $\mathsf{UC\text{-}Com}(P_1, P_2, a)$. The parties perform the following steps:

1. For every commitment to a string a of length n, P_1 generates n commitments to 0 and n commitments to 1. P_1 interacts with the token sent to it by P_2 and obtains signatures on these $2n$ commitments. In order to commit to the i^{th} bit of a string a (denoted by a_i), P_1 selects a commitment to either 0 or

1 whose signature it had obtained from the device sent by P_2 (depending on what a_i is).

- We note that P_1 cannot give the hardware token commitments to the bits of a alone and obtain the signatures on these commitments. Doing this would allow P_2's hardware token to perform a selective failure attack. In other words, P_2's hardware could be programmed to respond and output signatures only if some condition is satisfied by the input string a (e.g., all its bits are 0). Thus if P_1 still continues with the protocol, P_2 gains some non-trivial information about a. Hence, P_1 obtains signatures on n commitments to 0 and n commitments to 1 and then selects commitments (and their signatures) according to the string a. This makes sure that the interaction of P_1 with the hardware token is independent of the actual input a.

Let $B_i = \mathsf{Com}(a_i)$ and let the signature obtained by P_1 from the device on this commitment be $\sigma_i = \sigma_{PK}(B_i)$. P_1 now computes a commitment to σ_i for all $1 \leq i \leq n$ denoted by $C_i = \mathsf{Com}(\sigma_i)$.

Let $Com_i = (B_i, C_i)$. Now $A = \mathsf{COM}(a) = \{Com_1, Com_2, ..., Com_n\}$ (in other words, A is the collection of commitments to the bits of a and commitments to the obtained signatures on these commitments). P_1 sends A to P_2.

- Note here that P_1 does not send the obtained signatures directly to P_2, but instead sends a commitment to these signatures. This is because the signatures could have been maliciously generated by the hardware token created by P_2 to leak some information about a.

2. Let w be a witness to the NP statement that for all i, C_i is a commitment to a valid signature of B_i under P_2's public key PK and that B_i is a valid commitment to a bit. More formally, w is a witness to the following NP statement: "L: For all i,
   – There exists a valid opening of B_i to a bit a_i under the commitment scheme $\mathsf{Com}(\cdot)$
   – There exists a valid opening of C_i to a string σ_i under the commitment scheme $\mathsf{Com}(\cdot)$ such that $\mathsf{Verify}(PK, B_i, \sigma_i) = 1$."

   P_1 picks $l(k)$ random pairs $\{(w_0^1, w_1^1), (w_0^2, w_1^2), \cdots, (w_0^{l(k)}, w_1^{l(k)})\}$ ($l(k)$ is a super-logarithmic function in security parameter k) such that for all $1 \leq t \leq l(k), w_0^t \oplus w_1^t = w$. P_1 sends commitments to these $l(k)$ pairs. In other words, P_1 sends $\mathsf{Com}(w_0^t), \mathsf{Com}(w_1^t)$ for all t.
3. P_2 picks $l(k)$ random bits $\{q_1, q_2, \cdots, q_{l(k)}\}$ and sends it to P_1.
4. P_1 opens the commitment $\mathsf{Com}(w_{q_t}^t)$ for all t by sending $\mathsf{Open}(\mathsf{Com}(w_{q_t}^t))$.
5. P_1 now gives a concurrent zero-knowledge proof ([PRS02]) that w is a witness to statement L being true and that $w_0^t \oplus w_1^t = w$ for all t.
   - We use the specific concurrent zero knowledge protocol of [PRS02] as we require indistinguishability of simulated proof from real proof when the NP statement being proven is not fixed, but publicly predictable given the history of the protocol (as noted in [BPS06]).

Decommitment phase. The parties perform the following steps:

1. P_1 sends P_2 the string that was initially committed to. In particular, P_1 sends a to P_2.
   - Note that P_1 does not send the actual opening to the commitment. P_1 will later prove in zero knowledge that a was the string committed to in the commitment phase. This is to allow equivocation of the commitment by the simulator during protocol simulation.
2. We denote the following steps by the protocol $\mathsf{HardwareZK}(P_1, P_2, a)$:
   (a) P_2 picks a string R uniformly at random from $\{0,1\}^{p(k)}$ and executes the commitment protocol $\mathsf{UC\text{-}Com}(P_2, P_1, R)$.
   - P_1 will prove in zero knowledge that a was the string committed to in the commitment phase. Since we require straight-line simulation, the simulator would have to know in advance the challenge queries made by P_2 in this zero knowledge proof. Hence before this zero knowledge proof is given, P_2 commits to his randomness R using the UC-secure commitment protocol.
   - We note that the decommitment to R need not be equivocable by the UC-simulator and hence we avoid having to use the UC-secure decommitment protocol itself, which would have lead to circularity!

   (b) P_1 gives a standard zero knowledge proof that a is the string that was committed to in the commitment phase of the protocol. The randomness used by P_2 in this zero knowledge proof is R and along with every message sent in the zero knowledge protocol, P_2 proves using a standard zero knowledge proof that the message uses randomness according to the string R.

   Denote by R_i and a_i the i^{th} bits of R and a respectively. More formally, the statement P_1 proves to P_2 is "There exists randomness such that for all $i, B_i = \mathsf{Com}(a_i)$, where B_i is as sent in the commitment phase." Let the value $\mathsf{COM}(R)$ sent during $\mathsf{UC\text{-}Com}(P_2, P_1, R)$ be denoted by Z. Note that Z is of the form $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ where $X_i = \mathsf{Com}(R_i)$ and Y_i is a commitment to the signature of X_i under P_1's public key. The statement P_2 proves to P_1 is "There exists string R, such that
   – For all i, there exists an opening of X_i to R_i under the commitment scheme $\mathsf{Com}(\cdot)$
   – R was the randomness used to compute this message."

   (c) P_2 accepts the decommitment if and only if the proof given by P_1 was accepted.

5 Security Proofs

5.1 Description of Simulator

In order to prove UC security of the commitment functionality, we will need to construct a straight-line simulator that extracts the committed value in the commitment phase of the protocol and that can equivocate a commitment to a given value in the decommitment phase of the protocol. Below, we describe such a simulator that runs straight-line both while extracting the committed string when interacting with a committer P_1, as well as when equivocating a commitment to a receiver P_2.

Token Exchange phase. In this phase, before a party P_i begins a protocol with P_j, if P_i is honest then the UC-simulator S creates the program code for the token to be created by P_i and sent to P_j (according to the honest token creation protocol) and sends a copy of the program code to $\mathcal{F}_{wrap}$. If P_i is malicious, it creates the token by interacting with $\mathcal{F}_{Adv}$ as described before. We again note that the token creation can be done at any point before P_i begins a protocol with P_j.

Handling token queries. Whenever an adversarial party queries a token created by another adversarial party, the simulator S forwards the request to $\mathcal{F}_{wrap}$. When simulating the view during the adversary's interaction with a token created by an honest party, S generates the response according to the request by the adversarial party and the program code of the token.

For every pair of parties (P_i, P_j) such that $P_i \in \mathcal{M}$ and $P_j \in \mathcal{H}$, S creates a table T_{ij}. When a malicious party P_i queries the token of an honest party P_j, S stores the query in table T_{ij}. In other words, the simulator S builds a list of all the commitments (along with their openings) that the malicious party queries to a token created by an honest party (for getting a signature). We shall show below that no matter how the tokens of malicious parties are created, the malicious parties cannot obtain any information about the inputs of honest parties.

When a malicious party P_i queries the token of a malicious party P_j, S simply forwards the query to $\mathcal{F}_{wrap}$ and forwards the response received from $\mathcal{F}_{wrap}$ back to P_i. We note that these queries can only make black box calls to tokens of honest parties (as malicious tokens can be created only with black box calls to tokens of honest parties). Hence whatever information an adversary can obtain from this query, the adversary could have obtained itself by making a black box query to the token of an honest party. Hence querying this token gives no additional information to an adversary.

Case 1: Committer is corrupted

Commitment Phase: In this case, the simulator S executes the protocol honestly as a receiver in the commitment phase. In more detail:

1. Let $A = \mathsf{COM}(a) = \{Com_1, Com_2, ..., Com_n\}$ according to the commitment protocol described earlier. P_1 sends A to S (Of course, P_1 may not follow the protocol).
2. Let w be a witness to the NP statement that for all i, C_i is a commitment to a valid signature of B_i under P_2's public key PK and that B_i is a valid commitment to a bit.

P_1 picks $l(k)$ random pairs $\{(w_0^1, w_1^1), (w_0^2, w_1^2), \cdots, (w_0^{l(k)}, w_1^{l(k)})\}$ ($l(k)$ is a super-logarithmic function in security parameter k) such that for all $1 \leq t \leq l(k)$, $w_0^t \oplus w_1^t = w$. P_1 sends commitments to these $l(k)$ pairs. In other words, P_1 sends $\mathsf{Com}(w_0^t), \mathsf{Com}(w_1^t)$ for all t.

3. S picks $l(k)$ random bits $\{q_1, q_2, \cdots, q_{l(k)}\}$ and sends it to P_1.
4. P_1 opens the commitment $\mathsf{Com}(w_{q_t}^t)$ for all t by sending $\mathsf{Open}(\mathsf{Com}(w_{q_t}^t))$.
5. P_1 now gives a concurrent zero-knowledge proof (from [PRS02]) that w is a witness to statement L being true and that $w_0^t \oplus w_1^t = w$ for all t.

The simulator S accepts the commitment if it accepts the zero-knowledge proof. If the zero knowledge proof was accepted, S looks up the commitments to the bits of a (i.e., B_i) in the table T_{12}. Note that T_{12} contains a list of all commitments that were queried by P_1 to the token created by honest party P_2. If any of the commitments are not found, then the simulator aborts the simulation. We call this an *Extraction Abort*. By a reduction to the security of the underlying signature scheme, we prove in Lemma 1 that Extraction Abort occurs with negligible probability. If the simulator did not abort, this means that the commitments to the bits of a were queried by P_1 to the device. Hence, the simulator S has already recorded the openings to these commitments and can extract a by looking up the opening of all these commitments B_i's in the table T_{12}.

Decommitment Phase: S follows the decommitment protocol honestly as a receiver. In more detail:

1. P_1 sends S the string a that was initially committed to. Dishonest P_1 may cheat and send $a' \neq a$ to S.
2. S picks a string R uniformly at random from $\{0,1\}^{p(k)}$ and executes the commitment protocol $\mathsf{UC\text{-}Com}(S, P_1, R)$ honestly.
3. P_1 gives a zero knowledge proof that a' is the string that was committed to in the commitment phase of the protocol. The randomness used by S in this zero knowledge proof is R and along with every message sent in the zero knowledge protocol, S proves in zero knowledge that the message uses randomness according to the string R.
4. S accepts the decommitment if the proof given by P_1 was accepted. Upon accepting the decommitment, S checks if a' was the string that was initially committed to in the UC-commitment protocol. If this is not the case, then S aborts. We call this a *Decommit Abort*. We show in Lemma 2 that Decommit Abort occurs with negligible probability.

We note that when the committer is corrupted, the simulator (as the receiver) follows the protocol honestly during protocol simulation and hence the simulated protocol is identical to the real protocol.

Case 2: Receiver is corrupted

Commitment Phase: The UC-simulator does as follows:

1. S sets the string a to be a string whose all the bits are 0 and then sends $A = \mathsf{COM}(a) = \{Com_1, Com_2, ..., Com_n\}$ according to the commitment protocol described earlier.
2. Let w be a witness to the NP statement that for all i, C_i is a commitment to a valid signature of B_i under P_2's public key PK and that B_i is a valid commitment to a bit.
   S picks $l(k)$ random pairs $\{(w_0^1, w_1^1), (w_0^2, w_1^2), \cdots, (w_0^{l(k)}, w_1^{l(k)})\}$ ($l(k)$ is a super-logarithmic function in security parameter k) such that for all $1 \leq t \leq l(k)$, $w_0^t \oplus w_1^t = w$. S sends commitments to these $l(k)$ pairs. In other words, S sends $\mathsf{Com}(w_0^t), \mathsf{Com}(w_1^t)$ for all t.
3. P_2 sends challenge bits $\{q_1, q_2, \cdots, q_{l(k)}\}$ to S.
4. S opens the commitment $\mathsf{Com}(w_{q_t}^t)$ for all t by sending $\mathsf{Open}(\mathsf{Com}(w_{q_t}^t))$.
5. S now gives a concurrent zero-knowledge proof that w is a witness to statement L being true and that $w_0^t \oplus w_1^t = w$ for all t.

Decommitment Phase: The UC-simulator has to equivocate the commitment to some value a' in the decommitment phase. The simulator proceeds as follows:

1. S sends a' to P_2.
2. P_2 picks a string R of length $p(k)$ and executes the commitment protocol $\mathsf{UC\text{-}Com}(P_2, S, R)$. Again, P_2 may not execute the protocol honestly. If this commitment is accepted, the simulator looks up the commitments to the bits of R in the table T_{21}. If any of the commitments are not found, then the simulator does an extraction abort. Otherwise, the simulator has obtained R.
3. The simulator S now has to give a zero knowledge proof that a' is the string that was committed to in the commitment phase of the protocol. Now given R, all of P_2's messages in this zero knowledge proof protocol are deterministic.
   S internally runs the simulation of this zero knowledge protocol (using the simulator S_{zk} for the underlying zero knowledge protocol). It runs the simulation as the verifier in the protocol (using the messages according to randomness R). Note that S can do this by interacting with prover S_{zk} and generating all messages of the verifier using randomness R. S obtains the simulated transcript of this protocol. Let the messages sent by S in this transcript be denoted by $m_1^V, m_2^V, \cdots, m_d^V$ and let the messages sent by S_{zk} (as the prover) in this simulated transcript be $m_1^P, m_2^P, \cdots, m_d^P$.
4. S will "force" this transcript upon P_2. That is, S sends messages to the party P_2 according to the simulated zero knowledge protocol transcript. At step t of the zero knowledge protocol, it sends the message m_t^P to P_2 and expects to receive m_t^V as response .
   Party P_2 is forced to use the randomness R because P_2, along with every message sent in the zero knowledge protocol, has to prove in zero knowledge that the message uses randomness according to the string R. By the soundness property of this zero knowledge proof (given by P_2), if P_2 sends a message that is not according to randomness R, it will fail in the zero knowledge proof.

We show in the full version of the paper [CGS07] that the view of the adversary in the simulation and in the real protocol are computationally indistinguishable in the commitment as well as decommitment phase.

5.2 Abort Lemmas

Lemma 1. *(Extraction Abort)*
Let ϵ denote the probability with which the simulator S aborts the simulation in the commitment phase (say for some session t and some committer $P_i \in \mathcal{M}$ and receiver $P_j \in \mathcal{H}$). Then, ϵ is negligible in k.

Proof. Let s be the total number of commitment sessions in the protocol. Pick at random the t^{th} commitment session between parties P_i and P_j (with $P_i \in \mathcal{M}$ and $P_j \in \mathcal{H}$). We note that with probability $> \frac{\epsilon}{s}$, during the t^{th} session between malicious P_i and honest P_j, the simulator for the first time in the protocol aborted the simulation. This is the commitment session in the protocol that first terminates in an abort by the simulator. We now focus on this particular session between P_i and P_j.

In this commitment protocol, consider the point upto when P_i (after sending $\mathsf{COM}(a)$) gives a commitment to $l(k)$ random pairs of the form (w_0^t, w_1^t) with $w_0^t \oplus w_1^t = w$. Let this point in the protocol be denoted by λ. We note that the probability with which the simulator aborted the simulation for the first time at session t between P_i and P_j given the prefix of the protocol upto λ is still $> \frac{\epsilon}{s}$ (This probability includes the probability with which this prefix happens.). Now, S goes forward in the simulation with malicious P_i in this session. The simulator completes the simulation of this session between P_i and P_j (The simulator might have to simulate sessions between other parties before finishing the simulation of this particular session.). If the simulator runs into an Extraction Abort in some other commitment session, then the simulator simply aborts the simulation as in that case, the t^{th} session between P_i and P_j was not the first time the simulator had to do an Extraction Abort. Similarly, if the simulator runs into a Decommit Abort in some parallel session, then the simulator aborts the simulation in that case as well. If the dishonest party aborts or does not respond in some parallel session, the simulator aborts in that case as well. We note that the probability with which the simulator completes this commitment session between P_i and P_j and then has to do an extraction abort is $> \frac{\epsilon}{s}$.

Upon aborting the t^{th} session between P_i and P_j, the simulator rewinds the environment back to point λ in the protocol. Now, using fresh randomness the simulator simulates this session between P_i and P_j (once again simulating other parallel sessions if needed). The probability with which the simulator completes the simulation of this commitment session and then does an Extraction Abort (using the fresh randomness) is again $> \frac{\epsilon}{s}$. Hence, the probability with which the simulator will abort at the end of the t^{th} session between P_i and P_j in both executions is $> \frac{\epsilon^2}{s^2}$. The probability with which adversary P_i commits to random shares that do not exclusive-or to the witness and then succeeds in giving a false zero knowledge proof is negligible. This follows from the soundness of the concurrent zero-knowledge proof. The probability with which the simulator picked the same randomness in both simulations (and hence failed to extract the witness) is $\frac{1}{2^{l(k)}}$. Hence with probability $> [\frac{\epsilon^2}{s^2}(1 - \frac{1}{2^{l(k)}}) - g(k)]$ (where $l(k)$ is a super-logarithmic function in k and $g(k)$ is any negligible function in k), the simulator will extract a valid witness to the statement P_i was proving to P_j in the t^{th} session.

Since the simulator aborted at the end of this session, this means that there exists a commitment $B_f = \mathsf{Com}(a_f)$ made by P_i whose signature $\sigma_{PK_j}(B_f)$ was not queried by P_i to the device created by P_j. Note that the witness of the statement (which P_i was proving to P_j) contains signatures of all commitments made in that session and, in particular, it contains $\sigma_{PK_j}(B_f)$. Hence with probability $> \frac{\epsilon^2}{s^2} - negl(k)$, we get a forgery of a signature in the existential forgery security game with P_j's public verification key PK_j. From the security of the signature scheme, it follows that $\frac{\epsilon^2}{s^2}$ is negligible in the security parameter and hence ϵ is also negligible in k. □

Lemma 2. *(Decommit Abort)*
Let μ denote the probability with which the simulator S aborts the simulation in the decommitment phase (say for some session t and some committer $P_i \in \mathcal{M}$ and receiver $P_j \in \mathcal{H}$). Then, μ is negligible.

Proof. We shall first show that the protocol $\mathsf{HardwareZK}(P_i, S, a)$ is computationally sound in the stand-alone setting. Consider the zero-knowledge proof $\mathsf{HardwareZK}(P_i, S, a)$. The steps in this proof are as follows:

- S picks a string R uniformly at random from $\{0,1\}^{p(k)}$ and executes the commitment protocol $\mathsf{UC\text{-}Com}(S, P_i, R)$ honestly.
- P_i gives a standard zero knowledge proof that a' is the string that was committed to in the commitment phase of the protocol. The randomness used by S in this zero knowledge proof is R and along with every message sent in the zero knowledge protocol, S proves using a standard zero knowledge proof that the message uses randomness according to the string R.
- S accepts the decommitment if the proof given by P_i was accepted.

Through a sequence of hybrid arguments, we will now show that this protocol has computational soundness in the stand-alone setting.

Hybrid H_0: This hybrid is exactly the same as the above protocol.

Hybrid H_1: This hybrid is exactly the same as H_0 except that the simulator will give simulated zero knowledge proofs in the second step (even though it has a witness). Since this proof is zero knowledge in the stand-alone setting, we have that the simulated proof is computationally indistinguishable from the real proof and hence H_1 is computationally indistinguishable from H_0.

Hybrid H_2: Hybrid H_2 to H_4 deal with proving that the commitment scheme $\mathsf{UC\text{-}Com}$ is computationally hiding in the stand alone setting. Hybrid H_2 is exactly the same as H_1 except that the simulator replaces concurrent zero knowledge proof given in $\mathsf{UC\text{-}Com}(S, P_i, R)$ by a simulated zero knowledge proof. Note that we do not require the concurrency property of the zero knowledge proof here (as we are considering only the stand-alone setting). Hence, it follows from the zero knowledge property of this proof that H_2 is indistinguishable from H_1.

Hybrid H_3: This hybrid is exactly the same as H_2 except that the simulator replaces the commitments to input R in the first step of $\mathsf{UC\text{-}Com}(S, P_i, R)$ to commitments to a value R' (chosen independently at random). It follows from the computational hiding property of these commitments that H_3 is indistinguishable from H_2.

Hybrid H_4: In $\mathsf{UC\text{-}Com}(S, P_i, R)$, the simulator gave a commitment to R in the first step of the protocol. Let w_{old} be a witness to the NP statement that for all i, C_i is a commitment to a valid signature of B_i under P_2's public key PK and that B_i is a valid commitment to a bit. In this case B_i is a commitment to the i^{th} bit of R. The simulator then followed the rest of the protocol according to this commitment. In particular, in the next step of the commitment phase, the simulator committed to random shares w_0^t, w_1^t such that $w_0^t \oplus w_1^t = w_{old}$. Note that in H_3, the commitments B_i were changed to commitments to R'. Hence, we now have a new witness w_{new} that proves that C_i is a commitment to a valid signature of B_i under P_2's public key PK and that B_i is a valid commitment to a bit.

Hybrid H_4 is exactly the same as H_3 except that the simulator changes the commitments to shares of w_{old} (i.e., commitments to w_0^t, w_1^t) to shares such that they exclusive-OR to w_{new}. Note that these commitments are not used anywhere else in the protocol as the simulator uses simulated concurrent zero knowledge proofs in the commitment phase. From the computationally hiding property of the commitments it follows from a standard hybrid argument that H_4 is indistinguishable from H_3.

Hybrid H_5: This hybrid is exactly the same as H_4 except that the simulator replaces the simulated zero knowledge proof in the $\mathsf{UC\text{-}Com}(S, P_i, R)$ protocol to honest concurrent zero knowledge proof. Again since we are only considering the stand-alone setting, it follows from the zero knowledge property of this proof that H_5 is indistinguishable from H_4.

We note that the difference from H_0 to H_5 is that the commitment $\mathsf{UC\text{-}Com}$ (S, P_i, R) has been replaced by $\mathsf{UC\text{-}Com}(S, P_i, R')$. The simulator still uses simulated zero knowledge proof that messages sent as verifier in the zero knowledge proof are according to randomness R. We shall now argue that if an adversary P^* can violate the soundness of the proof system in Hybrid H_5, then we can construct an adversary p^* that will violate the soundness of the underlying standard zero knowledge proof. p^* will act as verifier V in the above simulated protocol with P^* and as prover p^* in the underlying standard zero knowledge proof with verifier v. p^* as verifier V will initially commit to a random value R to P^* using $\mathsf{UC\text{-}Com}(S, P_i, R)$. V will then forward messages that it receives from P^* to v as messages of the prover p^*. Upon receiving a message from verifier v, p^* will send this message (as verifier V) to P^* along with a simulated zero knowledge proof that the randomness used to construct this message is R' (chosen independently at random). Now, if P^* can violate the soundness of the proof in the simulated protocol, then p^* can violate the soundness of the underlying zero knowledge proof. Thus, the proof in the simulated protocol is sound. By the indistinguishability of Hybrid H_5 from H_0, it follows that the zero knowledge protocol $\mathsf{HardwareZK}(P_i, S, a)$ has computational soundness in the stand-alone setting.

Now, let s be the total number of decommitment sessions in the protocol. Pick at random the t^{th} session between parties P_i and P_j (with $P_i \in \mathcal{M}$ and $P_j \in \mathcal{H}$). We note that with probability $> \frac{\mu}{s}$, during the t^{th} session between malicious P_i and honest P_j, the simulator for the first time in the protocol does a decommit abort. We now focus on this particular session between P_i and P_j. In this decommitment protocol, the decommitter P_i sends value a' as the first message and then executes protocol $\mathsf{HardwareZK}(P_i, S, a)$ with the simulator. We showed stand-alone soundness of $\mathsf{HardwareZK}(P_i, S, a)$. Since soundness is composable, this implies that $\mathsf{HardwareZK}(P_i, S, a)$ is computationally sound in the concurrent setting. Hence, a dishonest decommitter can only decommit to the value initially committed to. We note that while simulating the t^{th} session between P_i and P_j, the simulator might have to simulate other sessions (commitment and decommitment). If the simulator runs into a Decommit Abort in some other session, then the simulator aborts the simulation since then the t^{th} session between P_i and P_j will not be the first time that the simulator does a Decommit Abort. We note that simulator (except with negligible probability) will not run into an Extraction Abort in a parallel session (as argued in Lemma 1). Hence, μ is negligible. □

References

BCNP04. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)

BPS06. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS, pp. 345–354 (2006)

Can01. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)

CDPW07a. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)

CDPW07b. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)

CF01. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

CGGM00. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC, pp. 235–244 (2000)

CGS07. Chandran, N., Goyal, V., Sahai, A.: New constructions for uc secure computation using tamper-proof hardware. Cryptology ePrint Archive (2007), http://eprint.iacr.org/2007/334

CKL06. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. J. Cryptology 19(2), 135–167 (2006)

CLOS02. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)

DNS98. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: STOC, pp. 409–418 (1998)

DNW07. Damgaard, I., Nielsen, J.B., Wichs, D.: Universally composable multiparty computation with partially isolated parties. Cryptology ePrint Archive (2007), http://eprint.iacr.org/2007/332

GL89. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC, pp. 25–32 (1989)

GMR88. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)

Gol01. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, Cambridge (2001)

Gol04. Goldreich, O.: Foundations of Cryptography: Basic Applications. Cambridge University Press, Cambridge (2004)

HILL99. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (1999)

HMQU05. Hofheinz, D., Müller-Quade, J., Unruh, D.: Universally composable zero-knowledge arguments and commitments from signature cards. In: 5th Central European Conference on Cryptology (2005), http://homepages.cwi.nl/~hofheinz/card.pdf

Kat07. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)

NY89. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC, pp. 33–43 (1989)

PRS02. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS, pp. 366–375 (2002)

Author Index