

Xuejia Lai
Kefei Chen (Eds.)

LNCS 4284

Advances in Cryptology – ASIACRYPT 2006

12th International Conference on the Theory
and Application of Cryptology and Information Security
Shanghai, China, December 2006, Proceedings



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Xuejia Lai Kefei Chen (Eds.)

Advances in Cryptology – ASIACRYPT 2006

12th International Conference on the Theory
and Application of Cryptology and Information Security
Shanghai, China, December 3-7, 2006
Proceedings

Volume Editors

Xuejia Lai
Dept. of Computer Science and Engineering
Shanghai Jiaotong University
800 Dong Chuan Road, Min Hang
Shanghai 200240, P.R. China
E-mail: lai-xj@cs.sjtu.edu.cn

Kefei Chen
Dept. of Computer Science and Engineering
Shanghai Jiaotong University
800 Dong Chuan Road, Min Hang
Shanghai 200240, P.R., China
E-mail: kfchen@sjtu.edu.cn

Library of Congress Control Number: 2006936348

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-49475-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-49475-1 Springer Berlin Heidelberg New York

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

Springer is a part of Springer Science+Business Media

springer.com

© IACR 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11935230 06/3142 5 4 3 2 1 0

Preface

ASIACRYPT 2006 was held in Shanghai, China, during December 3–7, 2006. This was the 12th annual ASIACRYPT conference, and was sponsored by the International Association for Cryptologic Research (IACR), in cooperation with the State Key Labs of Information Security, Chinese Academy of Sciences (LOIS), Lab for Cryptography and Information Security, Shanghai Jiaotong University (CIS/SJTU) and Natural Science Foundation of China (NSFC).

This year we received a record number of 314 submissions, of which 303 regular submissions were reviewed by 32 members of the Program Committee, with the help of 250 external referees. After a two-month review process, the Program Committee selected 30 papers for presentation. This volume of proceedings contains the revised version of the 30 selected papers. The IACR 2006 distinguished lecture by Ivan Damgaard was also in the program. The paper “Finding SHA-1 Characteristics” by Christophe De Cannière and Christian Rechberger received the best paper award.

The reviewing process was a challenging task, and we had to reject many good submissions that could have been accepted under normal circumstances. I am very grateful to Program Committee for their efforts to carry out this challenging task and to keep the high standard of ASIACRYPT conferences. We gratefully acknowledge our 250 external referees; without their help it would be infeasible to provide 1008 high-quality, often extensive, reviews. More importantly, I would like to thank all the authors who submitted their work to ASIACRYPT 2006.

This year submissions were processed using Web-based software iChair, and would like to thank Thomas Baigneres, Matthieu Finiasz and Serge Vaudenay for providing this valuable tool. I am grateful to Ruimin Shen for his generous and indispensable support and I would like to thank Changzhe Gao, Haining Lu and Jingjing Wu for the smooth operation of our Web-sites.

I would also like to thank the General Chair, Dingyi Pei, for organizing the conference and the Organization Chair, Kefei Chen, for taking over all the hard tasks and preparing these proceedings.

Last but not least, my thanks to all the participants of the ASIACRYPT 2006 conference.

September 2006

Xuejia Lai

ASIACRYPT 2006

December 3–7, 2006, Shanghai, China

Sponsored by

the International Association for Cryptologic Research (IACR)

in cooperation with

the State Key Labs of Information Security, Chinese Academy of Sciences
(LOIS)

and

Lab for Cryptography and Information Security, Shanghai Jiaotong University
(CIS/SJTU)

and

Natural Science Foundation of China (NSFC)

General Chair

Dingyi Pei Chinese Academy of Sciences, China

Program Chair

Xuejia Lai Shanghai Jiaotong University, China

Organization Chair

Kefei Chen Shanghai Jiaotong University, China

Program Committee

Paulo S.L.M. Barreto	University of Sao Paulo, Brazil
Mihir Bellare	U.C. San Diego, USA
Lily Chen	NIST, USA
Ed Dawson	Queensland University of Technology, Australia
Yvo G. Desmedt	University College London, UK
Giovanni Di Crescenzo	Telcordia Technologies, USA
Cunsheng Ding	Hong Kong University of Science and Technology, China
Henri Gilbert	France Telecom R&D, France
Guang Gong	University of Waterloo, Canada

Antoine Joux	DGA and University Versailles St-Quentin, France
Kwangjo Kim	ICU, Korea
Kaoru Kurosawa	Ibaraki University, Japan
Chi Sung Laih	National Cheng Kung University, Taiwan
Tanja Lange	Technical University of Denmark, Denmark
Arjen K. Lenstra	EPFL, Switzerland
Mulan Liu	Chinese Academy of Sciences, China
Wenbo Mao	HP Labs, China
Willi Meier	FHNW, Switzerland
Kaisa Nyberg	Helsinki University of Technology and Nokia, Finland
Kenny Paterson	Royal Holloway University of London, UK
David Pointcheval	CNRS/ENS, Paris, France
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Raphael C.W. Phan	Swinburne University of Technology, Malaysia
Phillip Rogaway	U.C. Davis, USA and Mah Fah Luang University, Thailand
Rei Safavi-Naini	University of Wollongong, Australia
Kouichi Sakurai	Kyushu University, Japan
Hovav Shacham	Weizmann Institute of Science, Israel
Serge Vaudenay	EPFL, Switzerland
Wenling Wu	LOIS, Chinese Academy of Sciences, China
Moti Yung	Columbia University, USA
Jianying Zhou	Institute for Infocomm Research, Singapore

External Referees

Michel Abdalla	Jiun-Ming Chen	Alex Dent
Joonsang Baek	Liqun Chen	Claus Diem
Thomas Baigneres	Benoit Chevallier-Mames	Christophe Doche
Lejla Batina	Jung-Hui Chiu	Yevgeniy Dodis
Amos Beimel	Kim-Kwang R. Choo	Ling Dong
Come Berbain	Yvonne Cliff	Jeroen Doumen
Daniel J. Bernstein	Martin Cochran	Orr Dunkelman
Olivier Billet	Scott Contini	Ratna Dutta
Peter Birkner	Jean-Sebastien Coron	Chun-I Fan
Alex Biryukov	Nicolas Courtois	Haining Fan
Alexandra Boldyreva	Jason Crampton	Reza Rezaeian Farashahi
Colin Boyd	Ricardo Dahab	Siamak
Ernie Brickell	Tanmoy Kanti Das	Fayyaz-Shahan-dashti
Sébastien Canard	Blandine Debraize	Serge Fehr
Ran Canetti	Cecile Delerabelle	Décio Luiz Gazzoni Filho
Zhengjun Cao	Yi Deng	Matthieu Finiasz
Dr Gary Carter	Yingpu Deng	Marc Fischlin

Eiichiro Fujisaki	Ari Juels	Juan Gonzalez Nieto
Kazuhide Fukushima	Pascal Junod	Juanma Gonzales Nieto
Jun Furukawa	Marcelo Kaihara	Svetla Nikova
Steven Galbraith	Yael Kalai	Yasuhiro Ohtaki
Marc Girault	Jonathan Katz	Dag Arne Osvik
Kenneth Giuliani	Pinhun Ke	Soyoung Park
Philippe Golle	John Kelsey	Matthew G. Parker
Zheng Gong	Shahram Khazaei	Sylvain Pasini
Eu-Jin Goh	Khoongming Khoo	Torben Pryds Pedersen
Jeroen van de Graaf	Aggelos Kiayias	Kun Peng
Jens Groth	Joe Kilian	Olivier Pereira
Lifeng Guo	Jongsung Kim	Ludovic Perret
Kishan Gupta	Eike Kiltz	Thomas Peyrin
Satoshi Hada	Takeshi Koshihara	Duong Hieu Phan
Safuat Hamdy	Lars Knudsen	Josef Pieprzyk
Helena Handschuh	Noboru Kunihiro	Benny Pinkas
Colm O hEigeartaigh	Simon Künzli	Angela Piper
Martin Hell	Wen-Chung Kuo	Weidong Qiu
Swee-Huay Heng	Hidenori Kuwakado	Tal Rabin
Yong-Sork Her	Joseph Lano	Leonid Reyzin
Javier Herranz	Sven Laur	Tom Ristenpart
Clemens Heuberger	John Malone Lee	Matt Robshaw
Alejandro Hevia	Reynald Lercier	Markus Rohe
Jason Hinek	Jung-Shian Li	Allen Roginsky
Shoichi Hirose	Pin Lin	Greg Rose
Martin Hirt	Xiangxue Li	Andy Rupp
Xuan Hong	Zhuowei Li	Ahmad-Reza Sadeghi
Nick Hopper	Benoit Libert	Palash Sarkar
Yoshiaki Hori	Yehuda Lindell	Louis Salvail
Honggang Hu	Yu Long	Werner Schindler
Lei Hu	Vadim Lyubashevsky	Katja Schmidt-Samoa
Po-Yi Huang	Mark Manulis	Berry Schoenmakers
Qianhong Huang	Anton Mityagin	Jasper Scholten
Xinyi Huang	Jean Monnerat	Jacob Schuldtt
Zhenjie Huang	Tal Moran	Gil Segev
J. J. Hwang	Yi Mu	SeongHan Shin
Jim Hughes	Sourav Mukhopadhyay	Tom Shrimpton
Russell Impagliazzo	Michael Naehrig	Andrey Sidorenko
Yuval Ishai	Jorge Nakahara Jr.	Alice Silverberg
Kouichi Itoh	Mridul Nandy	Leonie Simpson
Tetsu Iwata	Yassir Nawaz	Jerome A. Solinas
Stanislaw Jarecki	Gregory Neven	Nigel Smart
Shaoquan Jiang	Lan Nguyen	Adam Smith
Kwon Jo	Phong Nguyen	Markus Stadler
Ellen Jochemsz	Jesper Buus Nielsen	Martijn Stam

Allan Steel
Till Stegers
Damien Stehle
Andreas Stein
John Steinberger
Marc Stevens
Doug Stinson
Chunhua Su
Makoto Sugita
Haipo Sun
Hung-Min Sun
Willy Susilo
Daisuke Suzuki
Gelareh Taban
Tsuyoshi Takagi
Jun-ichi Takeuchi
Keisuke Tanaka
Qiang Tang
Tomas Toft
Dongvu Tonien
Jacques Traore
Pim Tuyls

Wen-Guey Tzeng
Yoshifumi Ueshige
Ingrid Verbauwhede
Frederik Vercauteren
Damien Vergnaud
Eric Verheul
Martin Vuagnoux
Charlotte Vikkelsoe
Johan Wallén
Colin Walter
Chih-Hung Wang
Guilin Wang
Huaxiong Wang
Kunpeng Wang
Wang Peng
Shuhong Wang
Bogdan Warinschi
Brent Waters
Benne de Weger
Mi Wen
Jian Weng
Christopher Wolf

Stefan Wolf
Zheng Xu
Yacov Yacobi
Akihiro Yamamura
Bo-Yin Yang
Chung-Huang Yang
C. N. Yang
Lizhen Yang
Yiqun Lisa Yin
Nam Yul Yu
Bin Zhang
Lei Zhang
Rui Zhang
Wentao Zhang
Zhengfeng Zhang
Zhifang Zhang
Xianfeng Zhao
Yunlei Zhao
Dong Zheng
Sujing Zhou
Yongbin Zhou
Huafei Zhu

Table of Contents

Attacks on Hash Functions

Finding SHA-1 Characteristics: General Results and Applications	1
<i>Christophe De Cannière, Christian Rechberger</i>	
Improved Collision Search for SHA-0	21
<i>Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Jun Yajima, Noboru Kunihiro, Kazuo Ohta</i>	
Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions	37
<i>Scott Contini, Yiqun Lisa Yin</i>	

Stream Ciphers and Boolean Functions

New Guess-and-Determine Attack on the Self-Shrinking Generator	54
<i>Bin Zhang, Dengguo Feng</i>	
On the (In)security of Stream Ciphers Based on Arrays and Modular Addition	69
<i>Souradyuti Paul, Bart Preneel</i>	
Construction and Analysis of Boolean Functions of $2t + 1$ Variables with Maximum Algebraic Immunity	84
<i>Na Li, Wen-Feng Qi</i>	

Biometrics and ECC Computation

Secure Sketch for Biometric Templates	99
<i>Qiming Li, Yagiz Sutcu, Nasir Memon</i>	
The 2-Adic CM Method for Genus 2 Curves with Application to Cryptography	114
<i>P. Gaudry, T. Houtmann, D. Kohel, C. Ritzenthaler, A. Weng</i>	
Extending Scalar Multiplication Using Double Bases	130
<i>Roberto Avanzi, Vassil Dimitrov, Christophe Doche, Francesco Sica</i>	

ID-Based Schemes

HIBE With Short Public Parameters Without Random Oracle	145
<i>Sanjit Chatterjee, Palash Sarkar</i>	
Forward-Secure and Searchable Broadcast Encryption with Short Ciphertexts and Private Keys	161
<i>Nuttapong Attrapadung, Jun Furukawa, Hideki Imai</i>	
On the Generic Construction of Identity-Based Signatures with Additional Properties	178
<i>David Galindo, Javier Herranz, Eike Kiltz</i>	

Public-Key Schemes

On the Provable Security of an Efficient RSA-Based Pseudorandom Generator	194
<i>Ron Steinfeld, Josef Pieprzyk, Huaxiong Wang</i>	
On the Security of OAEP	210
<i>Alexandra Boldyreva, Marc Fischlin</i>	
Relationship Between Standard Model Plaintext Awareness and Message Hiding	226
<i>Isamu Teranishi, Wakaha Ogata</i>	

RSA and Factorization

On the Equivalence of RSA and Factoring Regarding Generic Ring Algorithms	241
<i>Gregor Leander, Andy Rupp</i>	
Trading One-Wayness Against Chosen-Ciphertext Security in Factoring-Based Encryption	252
<i>Pascal Paillier, Jorge L. Villar</i>	
A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants	267
<i>Ellen Jochemsz, Alexander May</i>	

Construction of Hash Function

Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding	283
<i>Donghoon Chang, Sangjin Lee, Mridul Nandi, Moti Yung</i>	

Multi-Property-Preserving Hash Domain Extension and the EMD Transform	299
<i>Mihir Bellare, Thomas Ristenpart</i>	

Combining Compression Functions and Block Cipher-Based Hash Functions	315
<i>Thomas Peyrin, Henri Gilbert, Frédéric Muller, Matt Robshaw</i>	

Protocols

A Scalable Password-Based Group Key Exchange Protocol in the Standard Model	332
<i>Michel Abdalla, David Pointcheval</i>	

A Weakness in Some Oblivious Transfer and Zero-Knowledge Protocols	348
<i>Ventzislav Nikov, Svetla Nikova, Bart Preneel</i>	

Almost Optimum Secret Sharing Schemes Secure Against Cheating for Arbitrary Secret Distribution	364
<i>Satoshi Obana, Toshinori Araki</i>	

Block Ciphers

KFC - The Crazy Feistel Cipher	380
<i>Thomas Baignères, Matthieu Finiasz</i>	

Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions	396
<i>Jacques Patarin, Valérie Nachev, Côme Berbain</i>	

New Cryptanalytic Results on IDEA	412
<i>Eli Biham, Orr Dunkelman, Nathan Keller</i>	

Signatures

New Approach for Selectively Convertible Undeniable Signature Schemes	428
<i>Kaoru Kurosawa, Tsuyoshi Takagi</i>	

Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures	444
<i>Jens Groth</i>	

Analysis of One Popular Group Signature Scheme	460
<i>Zhengjun Cao</i>	
Author Index	467

Finding SHA-1 Characteristics: General Results and Applications

Christophe De Cannière^{1,2} and Christian Rechberger¹

¹ Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A-8010 Graz, Austria

² Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
{Christophe.DeCanniere, Christian.Rechberger}@iaik.tugraz.at

Abstract. The most efficient collision attacks on members of the SHA family presented so far all use complex characteristics which were manually constructed by Wang *et al.* In this report, we describe a method to search for characteristics in an automatic way. This is particularly useful for multi-block attacks, and as a proof of concept, we give a two-block collision for 64-step SHA-1 based on a new characteristic. The highest number of steps for which a SHA-1 collision was published so far was 58. We also give a unified view on the expected work factor of a collision search and the needed degrees of freedom for the search, which facilitates optimization.

1 Introduction

Shortcut attacks on the collision resistance of hash functions are usually differential in nature. In the differential cryptanalysis of block ciphers, characteristics with arbitrary starting and ending differences spanning less than the full number of rounds and having a sufficient high probability allow key recovery attacks faster than brute force. This contrasts the situation in the case of collision attacks on hash functions. Here characteristics of sufficiently high probability need to start and end with chaining input and output difference being zero, injected differences (via the message input) are expected to cancel out themselves.

Members of the MD4 hash function family like the widely used SHA-1 mix simple building blocks like modular addition, 3-input bit-wise Boolean functions and bit-wise XOR, combine them to steps and iterate these steps many times. High probability characteristics which are needed for fast collision search attacks exploit situations where differences with respect to one operation propagate with high probability through other building blocks as well. As an example, an XOR difference in the most significant bit of a word propagates with probability one through a modular addition. The best characteristics for SHA-1 are constructed such that these and similar effects are maximized. However they do not fulfill the requirement of zero differences at the chaining inputs/outputs which makes them not directly usable for fast collision search attacks. Earlier work on SHA-1 [2,13]

therefore consider characteristics which fulfill this requirement at the cost of a less optimal probabilities.

However, the fact that an attacker has complete control over the message input, and thus control over the propagation of all differences in the first steps, gives more freedom in the choice of good characteristics. Roughly speaking, the probability of complex characteristics spanning the first steps which *connect* to a desired high probability characteristic does not affect the performance of a collision search. Hence, finding these complex *connecting characteristics* helps to improve the performance of collision search attacks. In the case of SHA-1, finding such characteristics made differential collision search attacks on the full SHA-1 possible in the first place. To reflect the fact that the desired characteristics to connect to have usually probability one in a linearized model of the hash function, they are referred to as *L-characteristics*. The connecting characteristics do not have this property, hence the name *NL-characteristics*.

So far, little is known about the construction of these connecting NL-characteristics. Wang *et al.* describe in their seminal paper [20] an approach which is based on following and manipulating differences *manually* [23] in combination with a great deal of experience and intuition. Follow-up work on SHA-1 [16] as well as on MD4 [9], MD5 [3,7,8,15] and SHA-0 [10] all build up on the characteristics given in the papers of Wang *et al.* [17,20,21,22]. The only exception is recent work by Schl affer and Oswald [14] on the conceptually much simpler MD4, where an algorithm for finding new characteristics given the same message difference as originally used by Wang *et al.* is reported. No one succeeded so far in showing a similar ability in the case of SHA-1. By employing a new method and using SHA-1 as an example, we show in this article that finding useful NL-characteristics is also possible in more complex hash functions.

As shown in informal presentations by Wang [18,19], the actual shape/design of these connecting NL-characteristics interacts with speed-up techniques at the final-search stage. These techniques are referred to as message modification techniques and little details about them in the context of SHA-1 are publicly known so far. To sum up, two important methods (finding connecting NL-characteristics and message modification) are not fully understood, but heavily affect the actual collision-search complexity. Therefore, it currently seems impossible to reason about the limits of these techniques, other than improving on the current results in an ad-hoc manner. Hence the need for automated search tools as the one presented in this paper.

Looking at the recent results of Wang *et al.* on SHA-1, we see that more degrees of freedom are needed for speedup-purposes. As mentioned in [18], message conditions and state variable conditions need to be fulfilled for that purpose. It is observed that “the available message space is tight”, which refers to the remaining degrees of freedom.

The new view we propose unifies finding complex characteristics and speeding up the final search phase. By calculating the expected number of collisions, given the degrees of freedom, we tackle questions related to optimization. If the goal is to

find *one* collision, why should the used method allow to find more than that? The new view gives an attacker the ability to exploit *all* available degrees of freedom.

The remainder of the paper is structured as follows. Subsequently we define some notation in Table 1. A short description of SHA-1 is given in Sect. 2. We tackle the core of the problem in Sect. 3, where we revisit the approach of finding collisions based on differential techniques. To do that, we generalize the concept of characteristics and introduce a new way to calculate the expected work to find a collision. Some examples are given there to illustrate the new concept. Based on that, in Sect. 4 we finally describe a way to automatically find the complex NL-characteristics needed. Also there we give examples which illustrate its behavior. As an application of the described technique, we give a two-block 64-step SHA-1 colliding message pair including all used characteristics in Sect. 5. Sect. 6 puts our contribution into the context of related and previous work. We conclude and survey future work in Sect. 7.

Table 1. Notation

notation	description
$X \oplus Y$	bit-wise XOR of X and Y
ΔX	difference with respect to XOR
$X + Y$	addition of X and Y modulo 2^{32}
δX	difference with respect to modular addition
X	arbitrary 32-bit word
x_i	value of the i-th bit
X^2	pair of words, shortcut for (X, X^*)
M_i	input message word i (32 bits)
W_i	expanded input message word t (32 bits)
$X \lll n$	bit-rotation of X by n positions to the left, $0 \leq n \leq 31$
$X \ggg n$	bit-rotation of X by n positions to the right, $0 \leq n \leq 31$
N	number of steps of the compression function

2 Short Introduction to SHA-1

SHA-1 [11], as most dedicated hash functions used today, is based on the design principles of MD4. First, the input message is padded and split into 512-bit message blocks. An 80-step compression function is then applied to each of these 512-bit message blocks. It has two types of inputs: a chaining input of 160 bits and a message input of 512 bits. Let $g(m, h)$ denote the compression function with message input m and chaining input h . The chaining input h_{n+1} for the next compression function is calculated by $h_n + g(m, h_n)$, called feed forward. The chaining variables for the first iteration are set to fixed values (referred to as IV). The result of the last call to the compression function is the hash of the message. The compression function basically consists of two parts: the message expansion and the state update transformation.

Message Expansion. In SHA-1, the message expansion is defined as follows. The message is represented by 16 32-bit words, denoted by M_i , with $0 \leq i \leq 15$. In the message expansion, this input is expanded linearly into 80 32-bit words W_i . The expanded message words W_i are defined as follows:

$$W_i = \begin{cases} M_i, & \text{for } 0 \leq i \leq 15, \\ (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1 & \text{for } 16 \leq i \leq 79. \end{cases} \quad (1)$$

State Update Transformation. The state update transformation starts by copying the chaining input into the five 32-bit state variables A, \dots, E , which are updated in 80 steps ($0, \dots, 79$) by using the word W_i and a round constant K_i in step i . A single step of the state update transformation is shown in Fig. 1. The function f in Fig. 1 depends on the step number: steps 0 to 19 (round 1)

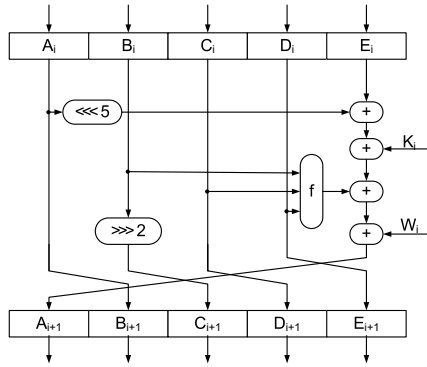


Fig. 1. One step of the state update transformation of SHA-1

use f_{IF} and steps 40 to 59 (round 3) use f_{MAJ} . The function f_{XOR} is applied in the remaining steps (round 2 and 4). The functions are defined as:

$$f_{IF}(B, C, D) = B \wedge C \oplus \overline{B} \wedge D \quad (2)$$

$$f_{MAJ}(B, C, D) = B \wedge C \oplus B \wedge D \oplus C \wedge D \quad (3)$$

$$f_{XOR}(B, C, D) = B \oplus C \oplus D. \quad (4)$$

Note that $B_i = A_{i-1}$, $C_i = A_{i-2} \ggg 2$, $D_i = A_{i-3} \ggg 2$, $E_i = A_{i-4} \ggg 2$. This also implies that the chaining inputs fill all A_j for $-4 \leq j \leq 0$. Thus it suffices to consider the state variable A , which we will for the remainder of this paper.

3 Collision Attacks Revisited

The objective of this paper is to develop a method to find SHA-1 characteristics which are suitable for collision attacks. However, in order to solve this problem,

we first have to determine exactly what ‘suitable’ means in this context. In this section, we will therefore consider collision attacks and characteristics in a general setting, and analyze how the choice of the characteristic affects the work factor of the attack.

3.1 How Dedicated Collision Attacks Work

If we are given an n -bit hash function whose output values are uniformly distributed and use it to hash an arbitrary pair of messages, then we expect the hash values to collide with a probability of 2^{-n} . Hence, without knowing anything about the internals of the hash function, we should be able to find a collision after trying out 2^n pairs. Since any set of 2^n pairs will do, this approach can be turned into a birthday attack requiring only $2^{n/2}$ hash evaluations.

Instead of testing arbitrary pairs, dedicated collision attacks try to use the internal structure of the hash function to locate a special subset of message pairs which (1) are considerably more likely to collide than random pairs, and (2) can efficiently be enumerated. A particularly effective way to construct such subsets is to restrict the search space to message pairs with a fixed difference. The goal is to pick these differences in such a way that they are likely to propagate through the hash function following a predefined differential characteristic which eventually ends in a zero difference (a collision).

As was observed in [4], the probability for this to happen can be increased by restricting the subset even further and imposing conditions not only on the differences but also on the *values* of specific (expanded) message bits. Moreover, since the internal variables of a hash function only depend on the message (and not on a secret key as for example in block ciphers), we can also restrict the set of message pairs by imposing conditions on the state variables. Depending on their position, however, these conditions might have a considerable impact on the efficiency to enumerate the messages fulfilling them. This important point is analyzed in detail in Sect. 3.3.

3.2 Generalized Characteristics

In order to reflect the fact that both the differences and the actual values of bits play a role in their attack, Wang *et al.* already extended the notion of differential characteristics by adding a sign to each non-zero bit difference (1 or -1). In this paper we generalize this concept even further by allowing characteristics to impose arbitrary conditions on the values of pairs of bits.

The conditions imposed by such a generalized characteristic on a particular pair of words X^2 will be denoted by ∇X . It will turn out to be convenient to represent ∇X as a set, containing the values for which the conditions are satisfied, for example

$$\nabla X = \{X^2 \mid x_7 \cdot x_7^* = 0, x_i = x_i^* \text{ for } 2 \leq i < 6, x_1 \neq x_1^*, \text{ and } x_0 = x_0^* = 0\}.$$

In order to write this in a more compact way, we will use the notation listed in Table 2. Using this convention, we can rewrite the example above as

$$\nabla X = [7?----x0].$$

Table 2. Possible conditions on a pair of bits

(x_i, x_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)	(x_i, x_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

A generalized characteristic for SHA-1 is then simply a pair of sequences $\nabla A_{-4}, \dots, \nabla A_N$ and $\nabla W_0, \dots, \nabla W_{N-1}$.

3.3 Work Factor and Probabilities

Let us now assume that we are given a complete characteristic for SHA-1, specified by $\nabla A_{-4}, \dots, \nabla A_N$ and $\nabla W_0, \dots, \nabla W_{N-1}$. Our goal is to estimate how much effort it would take to find a pair of messages which follows this characteristic, assuming a simple depth-first search algorithm which tries to determine the pairs of message words M_i^2 one by one starting from M_0^2 .

In order to estimate the work factor of this algorithm, we will compute the expected number of visited nodes in the search tree. But first we introduce some definitions.

Definition 1. *The message freedom $F_W(i)$ of a characteristic at step i is the number of ways to choose W_i^2 without violating any (linear) condition imposed on the expanded message, given fixed values W_j^2 for $0 \leq j < i$.*

We note that since the expanded message in SHA-1 is completely determined by the first 16 words, we always have $F_W(i) = 1$ for $i \geq 16$.

Definition 2. *The uncontrolled probability $P_u(i)$ of a characteristic at step i is the probability that the output A_{i+1}^2 of step i follows the characteristic, given that all input pairs do as well, i.e.,*

$$P_u(i) = P(A_{i+1}^2 \in \nabla A_{i+1} \mid A_{i-j}^2 \in \nabla A_{i-j} \text{ for } 0 \leq j < 5, \text{ and } W_i^2 \in \nabla W_i) .$$

Definition 3. *The controlled probability $P_c(i)$ of a characteristic at step i is the probability that there exists at least one pair of message words W_i^2 following the characteristic, such that the output A_{i+1}^2 of step i follows the characteristic, given that all other input pairs do as well, i.e.,*

$$P_c(i) = P(\exists W_i^2 \in \nabla W_i : A_{i+1}^2 \in \nabla A_{i+1} \mid A_{i-j}^2 \in \nabla A_{i-j} \text{ for } 0 \leq j < 5) .$$

With the definitions above, we can now easily express the number of nodes $N_s(i)$ visited at each step of the compression function during the collision search. Taking into account that the average number of children of a node at step i is

$F_W(i) \cdot P_u(i)$, that only a fraction $P_c(i)$ of the nodes at step i have any children at all, and that the search stops as soon as step N is reached, we can derive the following recursive relation:







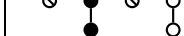

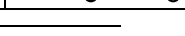
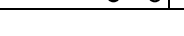
$$N_s(i) = \begin{cases} 1 & \text{if } i = N, \\ \max \{N_s(i+1) \cdot F_W(i)^{-1} \cdot P_u^{-1}(i), P_c^{-1}(i)\} & \text{if } i < N. \end{cases}$$

The total work factor is then given by

$$N_w = \sum_{i=1}^N N_s(i).$$

In order to understand what the different quantities defined above represent, it might be helpful to walk through a small example. Table 3 shows two hypothetical search trees with corresponding values of F_W , P_u , and P_c . The nodes which are visited by the search algorithm, and hence contribute to the complexity of the collision search, are filled. Note that the values of $P_c(i)$ do not always influence the complexity of the attack. The trees in Table 3, however, are examples where they do.

Table 3. How P_c affects the search tree

i	tree ^a	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$	i	tree	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$
0:		4	1/2	1	1	0:		4	1/2	1	1
1:		4	1/2	1	1	1:		4	1/2	1/2	2
2:		1	1/2	1/2	2	2:		1	1/2	1/2	2
3:		1	1	1	1	3:		1	1	1	1
4:		1	1	1	1	4:		1	1	1	1

^a Both \circ and \bullet represent values of W_{i-1}^2 which lead to a consistent A_i^2 ; the nodes visited by the search algorithm are filled. Inconsistent values are denoted by \ominus .

Let us now illustrate the previous concepts with two examples on 64-step SHA-1. In the first example, shown in Table 4, we consider a generalized characteristic which does not impose any conditions, except for a fixed IV value at the input of the compression function and a collision at the output. The values of $N_s(i)$ in the table tell us that the search algorithm is expected to traverse nearly the complete compression function 2^{160} times before finding a colliding pair (note that from here on all values listed in tables will be base 2 logarithms).

In the example of Table 5, we force the state variables and the expanded message words to follow a given differential characteristic starting from the output of the 16th step (i.e., A_{16}, \dots, E_{16}). How such differential characteristics can be found will be explained in Sect. 4. The most significant effect is that the five consecutive uncontrolled probabilities of 2^{-32} in the previous example move up to steps 11–15, where their effect on the number of nodes is completely neutralized by the degrees of freedom in the expanded message, resulting in a considerable reduction of the total work factor.

The examples above clearly show that small probabilities have a much larger impact on the work factor when they occur after step 16 (where $F_W(i) = 1$). Therefore, when constructing characteristics, we will in the first place try to optimize the probabilities in the second part of the compression function (steps 16 to $N-1$), even if this comes at the cost of a significant decrease of probabilities in the first part.

4 Constructing Characteristics

Having the necessary tools to estimate the work factor corresponding to any given generalized characteristic, we now turn to the problem of finding characteristics which minimize this work factor.

The search method presented in this section constructs characteristics by iteratively adding more conditions as long as it improves the work factor. During this process, two important tasks need to be performed: (1) determining when and where to add which condition, and (2) letting conditions propagate and avoiding inconsistent conditions. We first discuss the second problem.

4.1 Consistency and Propagation of Conditions

When analyzing the interaction of bit conditions imposed at the inputs and the outputs of a single step of the state update transformation, three situations can occur: (1) the conditions are inconsistent, (2) the conditions are consistent, and (3) the conditions are consistent, provided that a number of additional bit conditions are fulfilled as well (the conditions are said to propagate). This third case is illustrated in Table 6, where the conditions imposed on the expanded message words in the previous example propagate to the state variables. It should be noted that such consistency checks can be implemented in a very efficient way, thanks to the fact that bits at different bit positions only interact through the carries of the integer additions.

4.2 Determining Which Conditions to Add

In Sect. 3.3 we noted that conditions in a characteristic affect the work factor in very different ways depending on the step where they are enforced. This is also reflected in the procedure which we are about to propose: in order to determine where to add which conditions, we will proceed in a number of distinct stages, each of which tries to optimize a specific part of the characteristic.

Stage 1. As observed in Sect. 3.3, the work factor of the collision search algorithm is mainly determined by the shape of the characteristic after step 16. Hence, our first goal is to find a high probability differential characteristic, which can start with any difference in the state variables after step 16, but ends in a zero difference in the last step (later on, when we consider multi-block collisions, this constraint will be removed as well).

Table 4. Example 1, no conditions

i	∇A_i	∇W_i	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$
-4:	00001111010010111000011111000011					
-3:	010000001100100101010000111011000					
-2:	01100010111010101011001111111010					
-1:	1110111110011011010101110001001					
0:	01100111010001010010001100000001	????????????????????????????????	64	0.00	0.00	0.00
1:	????????????????????????????????	????????????????????????????????	64	0.00	0.00	0.00
				
12:	????????????????????????????????	????????????????????????????????	64	0.00	0.00	0.00
13:	????????????????????????????????	????????????????????????????????	64	0.00	0.00	0.00
14:	????????????????????????????????	????????????????????????????????	64	0.00	0.00	32.00
15:	????????????????????????????????	????????????????????????????????	64	0.00	0.00	96.00
16:	????????????????????????????????	????????????????????????????????	0	0.00	0.00	160.00
17:	????????????????????????????????	????????????????????????????????	0	0.00	0.00	160.00
				
59:	????????????????????????????????	????????????????????????????????	0	-32.00	0.00	160.00
60:	-----	????????????????????????????????	0	-32.00	0.00	128.00
61:	-----	????????????????????????????????	0	-32.00	0.00	96.00
62:	-----	????????????????????????????????	0	-32.00	0.00	64.00
63:	-----	????????????????????????????????	0	-32.00	0.00	32.00
64:	-----	-----				

Table 5. Example 2, less message freedom, better work factor by specifying a suitable message difference

i	∇A_i	∇W_i	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$
0:	01100111010001010010001100000001	-xx-----	32	0.00	0.00	0.00
1:	????????????????????????????????	xxx-----x-x-x-	32	0.00	0.00	0.00
				
7:	????????????????????????????????	-xx-----xx-x-	32	0.00	0.00	0.00
8:	????????????????????????????????	-xx-----x--xx	32	0.00	0.00	5.00
9:	????????????????????????????????	-x-----x----	32	0.00	0.00	37.00
10:	????????????????????????????????	xxx-----x--x-	32	0.00	0.00	69.00
11:	????????????????????????????????	-xx-----x-	32	-32.00	-29.00	101.00
12:	x-----	x-----x	32	-32.00	-31.00	101.00
13:	x-----	-----x----	32	-32.00	-31.00	101.00
14:	-----	-----xx	32	-32.00	-31.19	101.00
15:	x-----xx	-x-----x-x-x-	32	-32.00	-27.83	101.00
16:	-----x-	-----x-	0	-7.00	-4.00	101.00
17:	x-----x-	xxx-----x-x-x-	0	-7.00	-2.00	94.00
18:	-----x-	x-x-----x----	0	-5.00	-3.00	87.00
19:	-----x-	x-----x----	0	-4.00	-3.00	82.00
				
49:	-----x-	-----x----	0	-2.00	-1.00	7.00
50:	-----x-	-----x-	0	-3.00	-2.00	5.00
51:	-----	-----	0	-1.00	-1.00	2.00
52:	-----x-	-----	0	-1.00	-1.00	1.00
53:	-----x-	-----	0	0.00	0.00	0.00
54:	-----	-----	0	0.00	0.00	0.00
				
60:	-----	-----	0	0.00	0.00	0.00
61:	-----	-----	0	0.00	0.00	0.00
62:	-----	-----	0	0.00	0.00	0.00
63:	-----	-----	0	0.00	0.00	0.00
64:	-----	-----				

In general, the sparser a differential characteristic, the higher its probability, and in the case of the SHA family, it has been shown before that sparse characteristics can easily be found by linearizing all components of the state update

Table 6. Propagation of conditions in Example 2

i	∇A_i	∇W_i	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$
0:	01100111010001010010001100000001 ...	-xx-----	32	0.00	0.00	0.00
1:	??x-----	xxx-----x-x-x-	32	0.00	0.00	0.00
2:	????????????????????????????x-	--x-----x---xx	32	0.00	0.00	0.00
3:	??????????????????????????????	x-xx-----x-----	32	0.00	0.00	0.00
				

transformation, representing the resulting compression function as a linear code, and searching for low-weight vectors (see [5,12,13,20]).

Once a suitable differential characteristic is found for the linearized variant (called an L-characteristic), we determine the corresponding message difference, and impose it to our generalized characteristic. The differences in the state variables after step 16 are copied in the same way, except that we do not directly impose constraints to the most significant and the two least significant bits, but instead determine them by propagation. This will avoid inconsistencies caused in some cases by the nonlinear f -functions.

Stage 2. At this point, the largest part of the work factor is most likely concentrated in steps 12 to 16 (see e.g. Table 5), where the state variables, which are not constraint in any way in the previous steps, are suddenly forced to follow a fixed difference. In order to eliminate this bottleneck, we want to guide the state variables to the target difference by imposing conditions to the first steps as well.

Although the probability of this part of the characteristic is not as critical as before, we still want the differences to be reasonably sparse. Unfortunately, because of the high number of constraints (the message difference and both the differences at the input of the first step and at the output of step 16 are fixed already), suitable L-characteristics are extremely unlikely to exist in this case. In order to solve this problem, we will use a probabilistic algorithm which bears some resemblance to the algorithms used to find low-weight code words, but instead of feeding it with a linear code, we directly apply it to the unmodified (non-linear) compression function.

The basic idea of the algorithm is to randomly pick a bit position which is not restricted yet (i.e., a ‘?’-bit), impose a zero-difference at this position (a ‘-’-bit), and calculate how the condition propagates. This is repeated until all unrestricted bits have been eliminated, or until we run into an inconsistency, in which case we start again. The algorithm can be optimized in several ways, for example by also picking ‘x’-bits once they start to appear, guessing the sign of their differences (‘u’ or ‘n’), and backtracking if this does not lead to a solution. It turns out that inconsistencies are discovered considerably earlier this way.

An interesting property of the proposed procedure is that the sparser a characteristic, the higher the probability that it will be discovered. The number of trials before a consistent characteristic is found, is very hard to predict, though. Experiments show that this number can range from a few hundreds to several hundreds of thousands.

Stage 3. In the final stage, we try to further improve the work factor corresponding to the characteristic by performing local optimizations. To this end, we run through all bit positions of every state variable and every expanded message word, check which conditions can be added to improve the total work factor, and finally pick the position and corresponding condition which yields the largest gain. By repeating this many times, we can gradually improve the work factor. The example in Table 7 shows how our previous characteristic looks like after applying this greedy approach for a number of iterations.

An interesting issue here, is when to stop adding new conditions. In order to answer this question, we first notice that every additional condition reduces the size of the search tree, but at the same time lowers the expected number of surviving leaves at step N . In general, the work factor will improve as long as the search tree is reduced by a larger factor than the number of surviving leaves. At some point, however, the expected number of leaves will drop below one, meaning that message pairs which actually follow the characteristic are only expected to exist with a certain probability. This is not necessarily a problem if we are prepared to repeat the search for a number of different characteristics, and in fact, that is exactly how we constructed the second block of the 64-step collision presented in the next section. In this case, three different characteristics were used, the third of which is shown in Table 10 (notice that the expected number of characteristics needed to find one surviving leaf can directly be read from $N_s(0)$, in this example $2^{1.24} \approx 3$). Coming back to our original question, we

Table 7. Example 3, after adding conditions to minimize workfactor

i	∇A_i	∇W_i	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$
0:	01100111010001010010001100000001	0uu01010110011010000111101110101	0	0.00	0.00	0.00
1:	n0n0101010000001010100000101000	unn00001000110100010110111u1u0n0	0	0.00	0.00	0.00
2:	00u1unnnnnnnnnnnnnnnnnnnnnnn01u0	00n1110100110011111111011n1011uu	0	0.00	0.00	0.00
3:	1000101001100100100111u11100u111	n0um011000011010110011010u111100	0	0.00	0.00	0.00
4:	u000u01n11uu010u11u10100101010u0	un0n01101001000100010110n1u01uu	0	0.00	0.00	0.00
5:	n01001000n100011n1n000101uu0n010	uu1n1010111110011101110110n00u0	0	0.00	0.00	0.00
6:	01010011n0n0101u00100001000001100	10n100001111110000000000010011	0	0.00	0.00	0.00
7:	1011111unnnnnnnnnn100000nu101n10	1nu0100000010111001----001nu01u1	4	-1.00	0.00	0.00
8:	n1100110111000000101----00110nu00	0nu1101110111-----u0011nu	12	-8.00	0.00	0.00
9:	n01010010000111101110----n10111n	11u1100001111-----0u100111	11	-0.13	0.00	0.00
10:	n011010010111-----000000n0	nnn111101-----n1010u0	16	-4.00	-0.68	0.68
11:	u0110101011-----n1100100	1un1001-0-----0011u1	18	-6.00	-1.68	5.36
12:	u0010100101-----0-110001	u10110-0-0-----11000u	18	-11.00	-2.96	17.36
13:	u11100101110010-----0100000	0010010100000-----u00101	13	-4.00	-2.42	24.36
14:	01110011011111-----11000	1001000111111-----1001uu	11	-3.00	-2.00	33.36
15:	u1010110101-1-----1001uu	0n110-0-----n0n00n0	19	-10.14	-0.14	41.36
16:	1100011000000000-----110n0	1u0100101000-----u100100	0	0.00	0.00	50.22
17:	u000111011-----11u1	unn11101000000-----n0n10n1	0	-0.22	-0.21	50.22
18:	11101-----1001	n1u0--1-----01100101	0	-1.00	-0.48	50.00
19:	--0-----1u1	u00110-0-----n101011	0	-1.00	-0.54	49.00
20:	---0-----1--	10u00-1-1-----011100n	0	0.00	0.00	48.00
21:	-----u	00n--0-----nu01010	0	0.00	0.00	48.00
22:	-----	n1000-0-----010010u	0	-1.00	-1.00	48.00
60:	-----	-----0-----	0	0.00	0.00	0.00
61:	-----	-----1-0-----	0	0.00	0.00	0.00
62:	-----	-----1-1-----	0	0.00	0.00	0.00
63:	-----	-----0-----	0	0.00	0.00	0.00
64:	-----	-----				

can conclude that we should in principle continue adding conditions as long as the gain in work factor justifies the cost of generating additional characteristics.

5 Applications

To illustrate our method, we give a characteristic for a two-block collision of SHA-1 reduced to 64 steps with the standard IV. Note that for different initial chaining variables, different characteristics might be needed. This is in contrast to MD4 or MD5 where good characteristics are possible without having conditions on the chaining variables. In addition to the characteristic, we also give a message pair which follows the described characteristic and collides. Note that, to the best of our knowledge, not a single second block characteristics for SHA-0 or SHA-1 has been presented so far, neither in the literature nor in informal public talks. Hence the example we give is the first of its kind. Additionally, it is a collision for SHA-1 with the highest number of steps published so far (previously known collisions covered up to 58 steps).

5.1 On the Choice of the Message Difference

The choice of the message difference determines the high-probability characteristics L_1 that is followed in the later part of the compression function. This is illustrated in Fig. 2. In a first step, only '–' and 'x' conditions are needed, *i. e.* we only allow XOR-differences. The signs of the differences as well as some values of bits are determined in a later stage of the attack.

As previous work shows [5,12,13,20], it turns out that interleaving so-called local collisions (a disturbing and a set of correcting differences) is the best way to construct these high-probability characteristics in the case of SHA-1. It turns out that these characteristics are L-characteristics. In order to allow for a small work factor, we do not put restrictions on the output difference of the compression function. Thus, δh_1 will be nonzero. Good L-characteristics for variants of SHA-1 with other than 80 steps are usually shifted versions of each other. These effects have also been considered in previous work, thus we do not expand on this issue here. In order to turn such high probability characteristics, which actually describe a pseudo-near-collision, into a collision-producing characteristic, NL-characteristics are needed. As illustrated in Fig. 2, a first NL-characteristic (NL_1) is needed to connect from a zero-difference in the chaining variables to L_1 . After the feed-forward of the first block, we expect to have a modular difference $+d$ in the chaining variables.

However, this difference does not fit to the difference needed to directly connect to the same L-characteristic used in the first block. Regardless of that, we want to follow this L-characteristics in the second block again (with the exception of different signs for some differences). The reason is that we want to cancel out the expected low-weight difference after the last step of the second block with the difference that is fed forward. We require

$$\delta g(h_1, m_1) + \delta h_1 = 0.$$

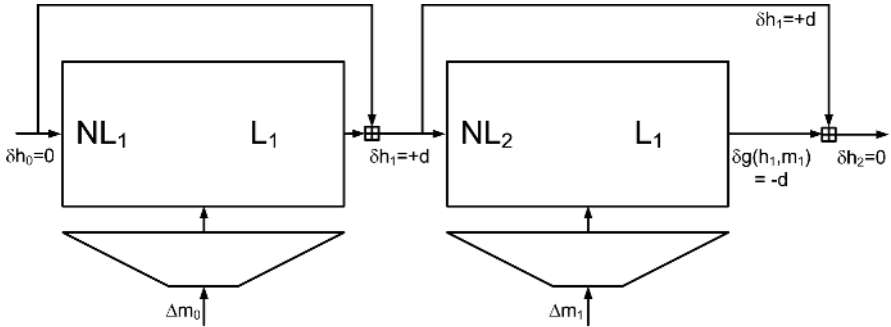


Fig. 2. Two-block approach to produce collisions

Thus, a new NL-characteristic (NL_2) for the second block is needed, taking into account the difference between δh_0 and δh_1 and the actual values at the chaining input of the second block. Note that with the ability to find these general NL-characteristics NL_1 and NL_2 , collision-producing characteristics covering more than two blocks do not improve the work factor.

In [20,22], examples for NL-characteristics are given which connect to a previously selected L-characteristic in the first block. It is commonly assumed that finding these NL-characteristics was based on experience and intuition, and done manually. Based on Sect. 3 and 4, we describe in the following an application for the *automatical* search for suitable NL-characteristics, which succeeds for the first and the second block.

5.2 A Two-Block Collision for 64-Step SHA-1

Herein we present a collision for 64-step SHA-1 using two message blocks. Table 9 and 10 detail the used characteristic for the first block and the second block respectively (see Sect. 3.2 for an explanation of the notation). Using our current (unoptimized) methods, we have an expected work factor of about 2^{35} compression function evaluations to find it. This compares favorably to the estimate of 2^{36} given in [20].

The number of nodes in the tree visited in the search, N_w , is given as the sum of all N_s in Tables 9 and 10. N_w relates to the expected work factor in the following way. We measured the cost of visiting a node in the search tree to be about 2^{-5} compression function evaluations. For that, we used as a means of comparison the SHA-1 implementation of OpenSSL 0.9.7g, which can do about 2^{19} compression functions per second on our PC. Note that the work factor for both blocks is lower than estimated. The reason is that carry differences in the last steps can be ignored and that the characteristic of the second block can be adjusted to allow additional deviations in the last steps of the first block.

In Table 8, we give the colliding messages. Note that we do not consider padding rules in our example, which would simply mean adding a common block to both messages after the collision. At this point we stress that this example

Table 8. Example of a 64-step collision using the standard IV

i	Message 1 (m_0), first block	Message 1 (m_1), second block
1–4	63DAEFDD 30A0D167 52EDCDA4 90012F5F	3B2AB4E1 AAD112EF 669C9BAE 5DEA4D14
5–8	0DB4DFB5 E5A3F9AB AE66EE56 12A5663F	1DBE220E AB46A5E0 96E2D937 F3E58B63
9–12	D0320F85 8505C67C 756336DA DFFF4DB9	BE594F1C BD63F044 50C42AA5 8B793546
13–16	596D6A95 0855F129 429A41B3 ED5AE1CD	A9B24128 816FD53A D1B663DC B615DD01
i	Message 2 (m_0^*), first block	Message 2 (m_1^*), second block
1–4	63DAEFDE 70A0D135 12EDCDE4 70012F0D	3B2AB4E2 EAD112BD 269C9BEE BDEA4D46
5–8	ADB4DFB5 65A3F9EB 8E66EE57 32A5665F	BDBE220E 2B46A5A0 B6E2D936 D3E58B03
9–12	50320F84 C505C63E B5633699 9FFF4D9B	3E594F1D FD63F006 90C42AE6 CB793564
13–16	596D6A96 4855F16B 829A41F0 2D5AE1EF	A9B2412B C16FD578 11B6639F 7615DD23
i	XOR-difference are the same for both blocks	
1–4	00000003 40000052 40000040 E0000052	00000003 40000052 40000040 E0000052
5–8	A0000000 80000040 20000001 20000060	A0000000 80000040 20000001 20000060
9–12	80000001 40000042 C0000043 40000022	80000001 40000042 C0000043 40000022
13–16	00000003 40000042 C0000043 C0000022	00000003 40000042 C0000043 C0000022
i	The colliding hash values	
1–5	A750337B 55FFFDDB C08DB36C 0C6CFD97	A12EFFE0

serves as a proof of concept for the unified approach to searching for complex characteristics and optimizing the characteristic for the final search phase. Hence it does not rule out other, probably more efficient ways to speed up the search for colliding pairs using the given characteristic.

6 Comparison with Previous Work

In order to put our contribution into perspective, we compare it with related previous work.

On finding suitable characteristics. In 1998, the pioneering work of Chabaud and Joux [4] resulted in a collision-search attack on an earlier version of SHA-1 (termed SHA-0). Their attack is based on L-characteristics they found. The Hamming weight of these characteristics (or a part of them) was used as a rough estimate of the attack complexity. However, the details depend on the positions of all differences. For each difference, the sign, the step in which it occurs, the bit-position within the word as well as its relative position to neighboring differences influence its impact on the attack complexity. A general and practical way to calculate this impact was described in Sect. 3.3.

In 2005, Rijmen and Oswald reported an attack on step-reduced SHA-1 [13], which is based on L-characteristics as well. Also the complexity of a collision search on SHA-0 was improved by Biham and Chen using the neutral-bit technique [1], and by Biham *et al.* using a multi-block approach [2]. Note that the attack on SHA-0 [2] employed four message blocks. Using the presented method of automatically finding complex characteristics, we eliminate the need for more than two blocks for an efficient collision-search attack.

Recent results of Wang *et al.* [20,22] describe further major improvements. By employing the multi-block technique as described in Sect. 5.1, together with the ability to manually find NL-characteristics, attack costs are improved by many orders of magnitude. As shown in Sect. 5, our method can be used to

automatically reach the same goal. This also answers the question left open in [16]. Since the NL-characteristic for the second block (NL_2) depends on the chosen message pair for the first block, this also prevents a manual search for new characteristics in the middle of a collision search.

The only related work which also aims for automatic search for complex characteristics is by Schl affer and Oswald [14] on MD4. Their method is very different from ours. It assumes a fixed differential behavior of the function f and limits carry extensions to only a few bit positions to reduce the search space. Thus it is not easy to extend it to more complex hash functions since these restrictions are too strict. Our method is not restricting anything, but is still practical.

On the cost of the final search. In previous work, the cost of the attack is further improved by a technique called message modification. The ideas developed in Sect. 3 and 4 can also be used for similar improvements. Both the originally published results by Wang *et al.* [20] as well as work by Sugita *et al.* [16] give rough estimates for the cost of message modification: 2^1 and 2^2 compression function evaluations (c_g), respectively. Sugita *et al.* also give a different trade-off. By using Gr obnerbasis-methods they reduce the number of trials significantly at the cost of increased message modification costs. Overall, this method does not lead to improvements in practice.

Note that for the recently announced but to the best of the authors knowledge unpublished improvements of the complexity of the collision search for full SHA-1 [18] (from 2^{69} to 2^{63}), no message modification costs are given, thus we lack comparability here.

Our approach can be seen as a trade-off towards very fast trials without the overhead of expensive message modification. As mentioned in Sect. 5.2, the cost of visiting one node in our search is only in the order of $2^{-5}c_g$. Note that the neutral-bit technique [1,2] can also be seen as a trade-off in this direction. However, as reported in [1], only a small fraction (one out of eight in the simpler case of SHA-0) of the trials conforms to a previously selected characteristic. Comparing the neutral-bit technique to our method, we observe two differences. Firstly, instead of a small fraction, we can be sure that every trial will conform to the characteristic we select. Secondly we don't rely on randomly generating message pairs which conform to a previously selected characteristic to bootstrap the final search. Instead we can exploit the available degrees of freedom in a sensible way.

On exploiting degrees of freedom. In Sect. 3.3, we described a method to calculate the expected number of collisions given a particular characteristic. Thus we can make a sensible use of degrees of freedom up to the point where we expect to find only one suitable message pair. In fact, also this distinguishes our approach from all previous work.

Table 9. Characteristic used for the first block of the 64-step collision

i	∇A_i	∇W_i	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$
-4:	00001111010010111000011111000011					
-3:	01000000110010010101000011011000					
-2:	011000101110101011100111111010					
-1:	111011111001101010101110001001					
0:	01100111010001010010001100000001	0110001110101011101111101111nu	0	0.00	0.00	1.07
1:	0000001110001111100010001001000n	0n1100001010000011010-010u1n01u1	1	0.00	0.00	1.07
2:	0n0010010100001010110-00011u0un0	0u01001011101101----11011n100100	4	-3.00	0.00	2.07
3:	1u10100001110010100-1un110unu110	unn10000000000-1001----10u0u11u1	5	-4.00	0.00	3.07
4:	1un0010110011110un1100-0n1n11nu1	n0n011011011010001-01111-10110101	2	-2.00	0.00	4.07
5:	n1u10110101un00010nu10u111000010	u1100101101000111111---1n101011	4	-4.00	0.00	4.07
6:	100u100u01111nu00u1110mu111u1un1	10u01110011001101-1-----10101n	7	-5.00	0.00	4.07
7:	nn1100101n1101011-1111-11u1001u0	00n100101010-101-----100nu11111	7	-5.00	0.00	6.07
8:	01110111001100u00010-0n11110u1u	u1010000001100--00---11-000010u	7	-6.00	0.00	8.07
9:	1nu0000101uuuu0u1110-1010n110n0	1n00010100000101-100-10-u111n0	4	-3.00	0.00	9.07
10:	1011000101n1111n111u-01n00un100	nu1101010110001-011---1u010un	6	-5.00	0.00	10.07
11:	nnnnnnnnnnnnnnnnnnnnnnnnnnnnnn	1u01111111111111-----0u10n01	9	-9.00	0.00	11.07
12:	0011010000001110110000110011000	010110010110110101101---1-0101nu	4	-3.00	0.00	11.07
13:	0100000000001000000111100-011000	0n001000010101-----n1010n1	11	-4.00	0.00	12.07
14:	100110001000011000-0-----0110101	nu00001010011-----1n11000u	11	-2.00	0.00	19.07
15:	1101101011111-1-----00010n	uu101101010-1-1-----1-1n011n1	11	-0.07	0.00	28.07
16:	11111100-----0-0111	1101001010100-----1010101u	0	-1.00	-1.00	39.00
17:	0000-----1-1111	1u0011100111-----1101011u0	0	-1.00	-0.99	38.00
18:	-----0-----01u	un00111011-0-0-----0n0011nu	0	0.00	0.00	37.00
19:	-----n-----	1u1100011111-----1un011n0	0	0.00	0.00	37.00
20:	-----n-----	n1101001100-----011000n	0	-1.00	-1.00	37.00
21:	-----n-----	1u1000110-1-0-----0u1000n0	0	-2.00	-2.00	36.00
22:	-----n-----	1n011010011-----0u0110n1	0	-2.00	-2.00	34.00
23:	-----n-----	0n10011011-----01111n0	0	-1.00	-1.00	32.00
24:	-----n-----	00101001-0-0-----001010n1	0	-1.00	-1.00	31.00
25:	-----n-----	0001110111-----1u100100	0	0.00	0.00	30.00
26:	-----n-----	n00010000-----0-1111n1	0	-1.00	-1.00	30.00
27:	-----n-----	n001111-1-1-----11101010	0	0.00	0.00	29.00
28:	-----n-----	u10111110-----11001n0	0	-1.00	-1.00	29.00
29:	-----n-----	n0011100-----1u110010	0	0.00	0.00	28.00
30:	-----n-----	001010-1-1-----101010110	0	-2.00	-2.00	28.00
31:	-----n-----	u0110101-----0u110111	0	0.00	0.00	26.00
32:	-----n-----	u101001-----011111010	0	-2.00	-2.00	26.00
33:	-----u-----	00010-1-0-----110n100000	0	0.00	0.00	24.00
34:	-----n-----	u011010-----0010101110	0	-2.00	-2.00	24.00
35:	-----n-----	101111-----010u1110001	0	0.00	0.00	22.00
36:	-----n-----	n111-1-1-----1010110u0	0	-1.00	-1.00	22.00
37:	-----n-----	110110-----100000000	0	0.00	0.00	21.00
38:	-----n-----	n1001-----010111110	0	0.00	0.00	21.00
39:	-----n-----	u11-0-1-----101101011	0	0.00	0.00	21.00
40:	-----n-----	01010-----01011100	0	0.00	0.00	21.00
41:	-----n-----	1011-----100100000	0	0.00	0.00	21.00
42:	-----n-----	00-0-0-----1001110001	0	0.00	0.00	21.00
43:	-----n-----	1101-----001111011	0	0.00	0.00	21.00
44:	-----n-----	011-----10010000	0	0.00	0.00	21.00
45:	-----n-----	1-1-0-----101111000	0	0.00	0.00	21.00
46:	-----n-----	110-----1011010010	0	0.00	0.00	21.00
47:	-----n-----	01-----101011000	0	0.00	0.00	21.00
48:	-----n-----	-0-0-----101100001	0	0.00	0.00	21.00
49:	-----n-----	10-----110101011	0	0.00	0.00	21.00
50:	-----n-----	0-----1010101n11	0	-1.00	-1.00	21.00
51:	-----n-----	0-1-----10u100011-	0	0.00	0.00	20.00
52:	-----n-----	1-----001000u11	0	-1.00	-1.00	20.00
53:	-----n-----	-----110111n00u	0	-2.00	-2.00	19.00
54:	-----n-----	-----1-----u11011-u	0	-1.00	-1.00	17.00
55:	-----n-----	-----101010u00u	0	-1.00	-1.00	16.00
56:	-----n-----	-----011n10u-	0	-2.00	-1.91	15.00
57:	-----n-----	0-----u111000-u	0	-1.00	-1.00	13.00
58:	-----n-----	-----0-1010un1u-	0	-2.00	-1.83	12.00
59:	-----u-----	-----1n01n11u-	0	-2.00	-1.87	10.00
60:	-----n-----	-----u-11000xu-0	0	-2.00	-1.00	8.00
61:	-----n-----	-----000n01ux-	0	-2.00	-1.00	6.00
62:	-----n-----	-----1000n00n-x	0	-3.00	-1.89	4.00
63:	-----n-----	-----u-10010-n-n	0	-1.00	-1.00	1.00
64:	-----n-----	-----				

Table 10. Third characteristic used for the second block of the 64-step collision

i	∇A_i	∇W_i	F_W	$P_u(i)$	$P_c(i)$	$N_s(i)$
-4:	11110011111100010000010000n10011					
-3:	01101110111000001010001110011101					
-2:	11001011101100100011110111000100					
-1:	100101011110100100111001n110101					
0:	1010000000111101110010101101000	0011101100101010101101-0111000nu	1	0.00	0.00	1.24
1:	1111001001110010110010-10000n1nu	1n1010101101000--0100101u1n11u1	3	-3.00	0.00	2.24
2:	uu10001001100001000001nu01un01u0	0u1001101001110010011--11n101110	2	-2.00	0.00	2.24
3:	0u1001010111100000mnn01011u1nn	nu1110111101010010011010n0u0in0	0	0.00	0.00	2.24
4:	0nu1110110110n0010uuuu0uuuuu1u10	n0n11101101111100010001000001110	0	0.00	0.00	2.24
5:	1000111nu11u0001m11111100100001	u01010110100011010-00101-u100000	2	-1.00	0.00	2.24
6:	u11110un101n0u0111-1011n010u1010	10n1011011100-10110----10011011u	5	-2.00	0.00	3.24
7:	u001u11nn0101011100n--0u011n111	11u1001111001----00-0-10uu00011	6	-4.00	0.00	6.24
8:	1n010101001u01n10000-0-11000u011	u011110010110----0----1-001110n	9	-7.00	0.00	8.24
9:	01001u1n10100110100101-1-uu10100	1n11110101100-----u0001n0	12	-10.00	0.00	10.24
10:	uuuuuuuuuuuuuuuuuuuuu-1100u011	nu01000011000100-----n101nu	9	-8.00	0.00	12.24
11:	010011101111100011111un-0111100	1n00101101111001001----1n001u0	6	-6.00	0.00	13.24
12:	11000000101111111111111111u10	101010011011001001000--001010nn	3	-2.00	-1.00	13.24
13:	011000010111111111111-0110110n	1n000001011011111--n1110u0	8	-2.24	0.00	14.24
14:	010111110011010110-----010u0	uu01000110110-----1u0-11nn	12	-4.00	0.00	20.00
15:	01010010010000010-----00nu	un110110000-0-0-----1-0n000n1	11	-1.00	0.00	28.00
16:	001001001011-----10010	1100010100000-----1101001n	0	0.00	0.00	38.00
17:	100000-----1000	0n1101111101-----11-001u1	0	-1.00	-0.99	38.00
18:	-----0-----0u1	nn11101111-0-1-----0n0010nu	0	0.00	0.00	37.00
19:	-----n	0u1100011010-----1un000n1	0	-1.00	-1.00	37.00
20:	-0-----n	n0101010011-----11-10110n	0	-1.00	-1.00	36.00
21:	-----n	1u0001000-0-0-----0u1000n1	0	-1.00	-1.00	35.00
22:	-----n	0n010001010-----0u-011n1	0	-2.00	-2.00	34.00
23:	-----n	1n10010111-----00101n1	0	-1.00	-1.00	32.00
24:	-----n	11011111-0-1-----000101n1	0	-1.00	-1.00	31.00
25:	-----n	0010000100-----0u010000	0	0.00	0.00	30.00
26:	-----n	u10011101-----001000u0	0	-1.00	-1.00	30.00
27:	-----n	n100100-0-0-----01010001	0	0.00	0.00	29.00
28:	-----n	u11001101-----0-0-100n0	0	-1.00	-1.00	29.00
29:	-----n	n1111011-----1u110000	0	0.00	0.00	28.00
30:	-----n	100110-1-1-----00-00100	0	-2.00	-2.00	28.00
31:	-----u	u0000101-----1n000111	0	0.00	0.00	26.00
32:	-----n	u011010-----0001111100	0	-2.00	-2.00	26.00
33:	-----n	11111-0-0-----0-u100101	0	0.00	0.00	24.00
34:	-----u	u011010-----0-0000000	0	-2.00	-2.00	24.00
35:	-----u	100100-----01n011010	0	0.00	0.00	22.00
36:	-----n	n100-0-1-----0-1-11010n0	0	-1.00	-1.00	22.00
37:	-----n	010111-----100001001	0	0.00	0.00	21.00
38:	-----n	u0001-----0-0001101	0	0.00	0.00	21.00
39:	-----n	u00-0-0-----101010100	0	0.00	0.00	21.00
40:	-----n	11110-----010000101	0	0.00	0.00	21.00
41:	-----n	0011-----011010010	0	0.00	0.00	21.00
42:	-----n	00-1-0-----01-001100	0	0.00	0.00	21.00
43:	-----n	1010-----001111100	0	0.00	0.00	21.00
44:	-----n	000-----11-0011100	0	0.00	0.00	21.00
45:	-----n	0-1-0-----1-1100101	0	0.00	0.00	21.00
46:	-----n	000-----0--010010	0	0.00	0.00	21.00
47:	-----n	11-----001010101	0	0.00	0.00	21.00
48:	-----n	-0-0-----1100111001	0	0.00	0.00	21.00
49:	-----n	10-----0-0111110	0	0.00	0.00	21.00
50:	-----n	0-----11-1100n10	0	-1.00	-1.00	21.00
51:	-----n	1-0-----10u010110-	0	0.00	0.00	20.00
52:	-----n	1-----0000001u10	0	-1.00	-1.00	20.00
53:	-----n	-----011011n10u	0	-2.00	-2.00	19.00
54:	-----n	-1-----u-11011-u	0	-1.00	-1.00	17.00
55:	-----n	-----111011u01u	0	-1.00	-1.00	16.00
56:	-----n	-----01-00n10u-	0	-2.00	-1.91	15.00
57:	-----n	1-----u101111-u-	0	-1.00	-1.00	13.00
58:	-----n	-----10-00un0u-	0	-2.00	-1.83	12.00
59:	-----u	-----0m01u11u-	0	-2.00	-1.87	10.00
60:	-----u	-----n-0-111xu-0	0	-2.00	-1.00	8.00
61:	-----u	-----0100u01ux-	0	-2.00	-1.00	6.00
62:	-----u	-----0-u11n-x-	0	-3.00	-1.89	4.00
63:	-----u	-----n-10110-u-n-	0	-1.00	-1.00	1.00
64:	-----u	-----n-10110-u-n-	0	-1.00	-1.00	1.00

7 Conclusions and Future Work

We described, for the first time, a computer-implementable method to search for complex characteristics as needed in the effective cryptanalysis of hash functions of the MD4 family like SHA-1. As a proof of concept, we gave the characteristics needed for a 64-step two-block collision of SHA-1. Furthermore, for the first time an actual collision for 64-step SHA-1 is produced, with an expected work factor of 2^{35} compression function computations.

We also tackled issues like work factors or degrees of freedom and put them into a precise framework. Thus an optimal exploitation of available degrees of freedom gets possible for goals like fast collision search.

Future work includes optimization of the found characteristics for different final search strategies, or the application of the described technique to other hash functions. Given the increased design complexity of members of the SHA-2 family compared to SHA-1, an automatic approach as described in our article seems to be highly beneficial for the analysis of these hash functions.

Given the ability to automatically incorporate some differences from the chaining variables at the start of the compression function, applications such as meaningful collisions or speeding up techniques like herding attacks [6] are also future work.

Acknowledgements

We thank Florian Mendel and Vincent Rijmen for many insightful discussions, and the anonymous reviewers for their helpful comments. The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The first author is supported by the Austrian Science Fund (FWF) project P18138.

Disclaimer

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

References

1. E. Biham and R. Chen. Near-Collisions of SHA-0. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
2. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby. Collisions of SHA-0 and Reduced SHA-1. In R. Cramer, editor, *Advances in Cryptology - EURO-CRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 36–57. Springer, 2005.

3. J. Black, M. Cochran, and T. Highland. A Study of the MD5 Attacks: Insights and Improvements. In M. Robshaw, editor, *Proceedings of Fast Software Encryption - FSE 2006, Graz, Austria, March 15-17, 2006*, volume 4047 of *LNCS*, 2006. To appear.
4. F. Chabaud and A. Joux. Differential Collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.
5. C. S. Jutla and A. C. Patthak. Provably Good Codes for Hash Function Design. In R. Cramer, editor, *Proceedings of SAC 2006*, LNCS. Springer, 2006. to appear.
6. J. Kelsey and T. Kohno. Herding Hash Functions and the Nostradamus Attack. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *LNCS*, pages 183–200. Springer, 2005.
7. V. Klima. Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105, 2006. <http://eprint.iacr.org/>.
8. J. Liang and X. Lai. Improved Collision Attack on Hash Function MD5. Cryptology ePrint Archive, Report 2005/425, 2005. <http://eprint.iacr.org/>.
9. Y. Naito, Y. Sasaki, N. Kunihiro, and K. Ohta. Improved Collision Attack on MD4. Cryptology ePrint Archive, Report 2005/151, 2005. <http://eprint.iacr.org/>.
10. Y. Naito, Y. Sasaki, T. Shimoyama, J. Yajima, N. Kunihiro, and K. Ohta. Message Modification for Step 21-23 on SHA-0. Cryptology ePrint Archive, Report 2006/016, 2006. <http://eprint.iacr.org/>.
11. National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at <http://www.itl.nist.gov/fipspubs/>.
12. N. Pramstaller, C. Rechberger, and V. Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In N. P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *LNCS*, pages 78–95. Springer, 2005.
13. V. Rijmen and E. Oswald. Update on SHA-1. In A. Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.
14. M. Schl affer and E. Oswald. Searching for Differential Paths in MD4. In M. Robshaw, editor, *Proceedings of Fast Software Encryption - FSE 2006, Graz, Austria, March 15-17, 2006*, volume 4047 of *LNCS*, 2006. To appear.
15. M. Stevens. Fast Collision Attack on MD5. Cryptology ePrint Archive, Report 2006/104, 2006. <http://eprint.iacr.org/>.
16. M. Sugita, M. Kawazoe, and H. Imai. Gr obner Basis Based Cryptanalysis of SHA-1. Cryptology ePrint Archive, Report 2006/098, 2006. <http://eprint.iacr.org/>.
17. X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.
18. X. Wang, A. Yao, and F. Yao. Cryptanalysis of SHA-1. Presented at the Cryptographic Hash Workshop hosted by NIST, October 2005.
19. X. Wang, A. Yao, and F. Yao. New Collision Search for SHA-1, August 2005. Presented at rump session of CRYPTO 2005.

20. X. Wang, Y. L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
21. X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
22. X. Wang, H. Yu, and Y. L. Yin. Efficient Collision Search Attacks on SHA-0. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.
23. Y. L. Yin. Personal Communication, March 2006.

Improved Collision Search for SHA-0

Yusuke Naito¹, Yu Sasaki¹, Takeshi Shimoyama²,
Jun Yajima², Noboru Kunihiro¹, and Kazuo Ohta¹

¹ The University of Electro-Communications, Japan
{tolucky, yu339, kunihiro, ota}@ice.uec.ac.jp

² FUJITSU LABORATORIES LTD
{shimo, jyajima}@labs.fujitsu.com

Abstract. At CRYPTO2005, Xiaoyun Wang, Hongbo Yu and Yiqun Lisa Yin proposed a collision attack on SHA-0 that could generate a collision with complexity 2^{39} SHA-0 hash operations. Although the method of Wang et al. can find messages that satisfy the sufficient conditions in steps 1 to 20 by using message modification, it makes no mention of the message modifications needed to yield satisfaction of the sufficient conditions in steps 21 and onwards.

In this paper, first, we give sufficient conditions for the steps from step 21, and propose submarine modification as the message modification technique that will ensure satisfaction of the sufficient conditions from steps 21 to 24. Submarine modification is an extension of the multi-message modification used in collision attacks on the MD-family. Next, we point out that the sufficient conditions given by Wang et al. are not enough to generate a collision with high probability; we rectify this shortfall by introducing two new sufficient conditions. The combination of our newly found sufficient conditions and submarine modification allows us to generate a collision with complexity 2^{36} SHA-0 hash operations. At the end of this paper, we show the example of a collision generated by applying our proposals.

Keywords: SHA-0, Collision Attack, Message Modification, Sufficient Condition.

1 Introduction

SHA-0 is the hash function issued by NIST in 1993 [5]. All hash functions must hold 3 properties: Pre-image Resistance, Second Pre-image Resistance and Collision Resistance. Collision Resistance means that it is very hard to find x, y such that $x \neq y$ and $H(x) = H(y)$, where $H(\cdot)$ is any hash function. Collision Resistance is more difficult to keep than any other property. The Collision Resistance of SHA-0 was broken recently [2]. This paper uses the term Collision Attack to refer to attacks that break Collision Resistance.

The first collision attack on SHA-0 was proposed by F. Chabaud and A. Joux in 1998 [3]. They employed differential attack and used XOR as the differential. After that, E. Biham and R. Chen improved [3], and found near collisions [1]. Near collision means x, y such that $x \neq y$ and $H(x), H(y)$ differ only by a small

number of bits. At the rump session of CRYPTO2004, the first announcement of finding a collision of SHA-0 was made by A. Joux [4]. Details of this attack were presented in EUROCRYPT2005 by E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby and C. Lemuet [2]. In 2004, Wang proposed an independent collision attack method on SHA-0 [10,11]. Wang's method uses the differential attack approach in which numerical operations are used as the differential. Subsequently, X. Wang, H. Yu and Y. Lisa Yin proposed an improved version of Wang's attack [14]. This method has complexity of 2^{39} SHA-0 hash operations, and is the most efficient collision attack method proposed so far.

The method of Wang et al. can be divided into 2 phases. In the pre-computation phase, a differential path and conditions that indicate that a collision is possible are constructed. In this paper, we call these conditions "sufficient conditions". Sufficient conditions define the triggers for ending collision search. In the collision search phase, an input message satisfying all sufficient conditions is searched for. If this message is found, a collision can be generated. In this phase, message modification is used to efficiently find a message that satisfies the sufficient conditions.

According to Wang et al., in the case of SHA-0, a message satisfying sufficient conditions from steps 1 to 20 can be located efficiently by using message modification. The specification of SHA-0 states that the messages used in steps 1-16 are input messages, whereas messages used in steps after 16 are determined by message expansion as is defined by the specification of SHA-0. In the method of Wang et al., messages satisfying the sufficient conditions in steps 1-16 can, with probability 1, be generated by using message modification. Since steps 1-16 are not affected by the limitations placed on message expansion, it is possible to choose values of chaining variables to satisfy all sufficient conditions, and then calculate messages that can yield these chaining variables. Regarding the sufficient conditions in steps 17-20, if these conditions are not satisfied and message modification is executed, these sufficient conditions are satisfied with probability of almost 1. Since the steps from 17 are affected by message expansion, the message modification in steps after 16 proposed by Wang et al., is executed by generating the differential in the step not affected by message expansion. Since this differential (We call this differential "transmission differential") is transferred to subsequent steps, sufficient conditions are satisfied by the transferred differential. We call this method "transmission method". Without using these methods, the probability that a sufficient condition is satisfied in 1 time is $\frac{1}{2}$. For example, suppose there exists 1 condition in step i and the complexity to calculate all necessary operations up to step i is j steps. In this case, the number of steps needed to ensure the success of step i is $2j$ (on average). By using these methods, if the complexity of message modification is p steps, the number of steps needed to ensure the success of step i is $j + \frac{1}{2} \cdot p$ (on average). Since we choose message modification such that the complexity is $p < j$, message modification reduces the complexity by $j - \frac{1}{2} \cdot p$ steps. Therefore, we can efficiently locate a collision by using message modification. Note that message modification in the steps after 16 is particularly important in reducing the complexity of collision search.

Our Results

Our paper makes 2 contributions.

1st Result: Wang et al. have not proposed message modification to satisfy the sufficient conditions from step 21; their solution is random search. In this paper, we propose message modification for steps 21-24. We call this proposal “submarine modification”. It takes advantage of the ideas of multi-message modification for the MD-family (we call multi-message modification for the MD-family “cancel method”) and transmission method (Details are described below). Since the same discussion about the complexity of message modification made with regard to the proposal of Wang et al., discussed above, can be applied to submarine modification, submarine modification can more efficiently satisfy the sufficient conditions than random search. Since the structure of the MD-family or SHA-1 is very similar to that of SHA-0, submarine modification may also be applicable to those hash functions.

2nd Result: We show that the sufficient conditions given by Wang et al. are missing two conditions, and then describe the missing sufficient conditions.

From the second result, even if a message satisfying all sufficient conditions given by Wang et al. is found, collision search does not always succeed. Since their conditions are two short, their method will fail with probability $\frac{3}{4}$. We identify the two missing sufficient conditions and use them with our submarine modification proposal to search for a collision. Considering the fact that the number of sufficient conditions in steps 21-24 is 4 and given the complexity of submarine modification, a computer experiment finds that our method finds a collision with complexity 2^{36} SHA-0 hash operations. The PC used had a Pentium4 3.4GHZ CPU(OS: Linux 2.6.9 (Fedora Core 3, Red Hat 3.4.2), Compiler: gcc 3.4.2-i386). In the fastest case, a collision was found in 8 hours. The average time to find a collision was roughly 100 hours.

Overview of Our Main Idea: Submarine Modification

Submarine modification uses two ideas of message modifications, “transmission method” and “cancel method”. We can satisfy sufficient conditions for up to step 24 by using submarine modification.

“Transmission method” is the method that can satisfy sufficient conditions for up to step 21 of SHA-0 (Wang et al. apply transmission method to sufficient condition for steps 17-20. We confirm that transmission method is applicable to satisfy sufficient conditions for steps 17-21). Namely, transmission method can satisfy sufficient conditions for 5 steps from a start step of transmission.

“Cancel method” is the method that uses the idea of the local collision. The local collision is the method where we create a differential and offset the differential in within several. We construct the method that inputs differentials and offsets the effects of these differentials before step 16 such that the differential (we call this differential “latent differential”) appears again from step 17 due to message expansion after the differential is offset. Differentials don’t appear for steps between the step where the differential offsets and the step where the

latent differential appears. We call these steps “latent period”. We denote the number of steps in latent period after step 17 as t . Influence of differentials created before the step where the latent differential appears does not occur. Cancel method is the method with which the sufficient condition for the step where the latent differential appears is satisfied by using the latent differential. We use the idea of cancel method in order to allow the start step of transmission to locate between step 17 to step 19. Note that cancel method itself does not use transmission of the latent differential.

The method that we propose in this paper satisfies sufficient conditions for up to step 24. If we use transmission method to satisfy sufficient conditions for up to step 24, we need to extend the range where the transmission differential can be started from step 16 to step 19. We can realize it by using the idea of cancel method. Since maximum number of latent period after step 17 for SHA-0 is $t = 3$, we can extend the range of the start step of transmission from step 16 to step 19 if we adopt the transmission differential as the latent differential. The latent differential can be created by using cancel method. Since there exists no influence for satisfied sufficient conditions in latent period by using cancel method, and we can satisfy sufficient conditions for 5 steps from the start step of transmission by applying transmission method. Since this method takes advantage of the differentials whose local effects are cancelled in the earlier steps, we call this message modification technique “submarine modification”.

2 Structure of SHA-0[5]

SHA-0 is a hash function issued by NIST in 1993. SHA-0 has the Merkle-Damgård structure, therefore, it repeatedly applies a compression function. SHA-0 input is an arbitrary length message M , and SHA-0 output is 160 bit data $H(M)$. If the length of the input message is not a multiple of 512, the message is padded to realized a multiple of 512 bits. The padding process is $M^* = M||10\dots0$. First, 1 is added, and then as many 0’s as are needed. Padded message M^* is divided into several messages M_i each 512 bits long ($M^* = (M_1||M_2||\dots||M_n)$). These divided messages are input to the compression function.

$$h_1 = \text{compress}(M_1, IV) \rightarrow h_2 = \text{compress}(M_2, h_1) \rightarrow \dots \rightarrow h_n = \text{compress}(M_n, h_{n-1})$$

$$H(M) = h_n$$

In this paper, we call the calculation performed in a single run of the compression function 1 block. IV in the above expression is defined as $(a_0, b_0, c_0, d_0, e_0) = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476, 0xc3d2e1f0)$. We next explain the structure of the compression function of SHA-0. All calculations in this are 32-bit. In this paper, we exclude the description of “mod 2^{32} ”.

Procedure 1. Divide the input message M_j into 32 bit messages m_0, m_1, \dots, m_{15} .

Procedure 2. Calculate m_{16} to m_{79} by $m_i = m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}$

Procedure 3. Calculate chaining variables a_i, b_i, c_i, d_i, e_i in step i by the following procedures.

$$a_i = (a_{i-1} \lll 5) + f(b_{i-1}, c_{i-1}, d_{i-1}) + e_{i-1} + m_{i-1} + k_{i-1},$$

$$b_i = a_{i-1}, c_i = b_{i-1} \lll 30, d_i = c_{i-1}, e_i = d_{i-1}$$

“ $\lll j$ ” denotes left cyclic shift by j bits. Repeat this process 80 times. Initial values a_0, b_0, c_0, d_0, e_0 for the compression function of the first block are IV . a_0, b_0, c_0, d_0, e_0 for the compression function from the second block are the output values of the previous block. Steps 1-20 are called the first round. Steps 21-40, 41-60, and 61-80 are the second, third, and fourth rounds, respectively, k_i is a constant defined in each round. Function f is a boolean function defined in each round. The specifications of k_i and f are shown in Table 1.

Table 1. Function f and Constants k in SHA-0

round	function f	constant k_i
1	$(b \wedge c) \vee (\neg b \wedge d)$	0x5a827999
2	$b \oplus c \oplus d$	0x6ed9eba1
3	$(b \wedge c) \vee (c \wedge d) \vee (d \wedge b)$	0x8f1bbcdc
4	$b \oplus c \oplus d$	0xca62c1d6

Procedure 4. $(a_0 + a_{80}, b_0 + b_{80}, c_0 + c_{80}, d_0 + d_{80}, e_0 + e_{80})$ is the output of the compression function.

3 Collision Attack by Wang et al.[8,9,14,15]

The method of Wang et al. is based on differential attack which uses subtraction as the differential. If a collision is found on hash function $H(\cdot)$, that is, M, M' such that $H(M) = H(M'), M \neq M'$ is found, the differential values of M and $H(M)$ become $\Delta M = M' - M \neq 0$, $\Delta H(M, M') = H(M') - H(M) = 0$. Let x and x' be certain values. We write $x' - x$ as Δx , and we call Δx the differential value of x . Since the differential value of input message $\Delta M \neq 0$, differential values of the chaining variables of the hash function are not 0.

The method of Wang et al. first notes differential values. It determines the differential values of the chaining variables and the differential value of the input message so that the output differential value of hash function $\Delta H(M, M')$ becomes $\Delta H(M, M') = 0$ and the differential value of the input message becomes $\Delta M \neq 0$. However, even if we find a pair of messages M, M' that satisfy ΔM , the output differential value is not always $H(M') - H(M) = 0$. This can happen since the differentials of chaining values from M and M' do not always satisfy the differential values of the chaining variables. Therefore, we need to set conditions for satisfying the differential values of the chaining variables. We call

these conditions “sufficient conditions”. These procedures (deciding the differential value of the input message, differential values of the chaining variables and sufficient conditions) are pre-computations.

We start collision search by using the differential value of input message ΔM and sufficient conditions decided in the pre-computation phase. First, we search for message M satisfying all sufficient conditions. Next, we calculate $M' = M + \Delta M$. M and M' thus become collision messages, that is, $H(M) = H(M')$. In order to efficiently locate a message that satisfies all sufficient conditions, message modification can be used.

3.1 Message Modification for SHA-0 and MD-Family

First, we explain message modification for SHA-0, and clarify the range wherein message modification can be applied. Next, since we use the idea of cancel method, which is originally proposed for MD-family, as part of the proposed submarine modification, we explain the procedures of cancel method.

Message Modification for SHA-0 [14]

Message modification for SHA-0 can generate messages satisfying all sufficient conditions in steps 1-16 with probability of 1. This procedure is shown below.

- **Message Modification for step i ($1 \leq i \leq 16$):**
 1. Generate a_i satisfying all sufficient conditions for a_i .
 2. Calculate $m_{i-1} \leftarrow a_i - (a_{i-1} \lll 5) - f(b_{i-1}, c_{i-1}, d_{i-1}) - e_{i-1} - k_{i-1}$.

Transmission method was proposed by Wang et al as follows. These modifications are executed when sufficient conditions are checked and found to be not satisfied. In message modification for steps 17-20, differentials are generated in order to create a differential on a bit where the sufficient condition that we want to satisfy exists. From the specification of SHA-0, since we can freely choose messages only for steps 1-16, we input the differential on the message used in up to step 16. We then transfer this differential to step 17, which yields the differentials that impact the targeted bits in the subsequent steps.

Multi-message Modification for MD-Family [8,9]

Multi-message modification for the MD-family (which we call *cancel method*) involves modifying messages to satisfy the sufficient conditions from step 17 of the MD-family. In cancel method, differentials are input in steps which are not affected by message expansion, and then cancel the impact of those differentials. The differentials that are input appear in step 17 and later steps due to message expansion, and this leads to satisfaction of the sufficient conditions. Cancel method does not use the technique where the latent differential transfers.

3.2 Collision Search for SHA-0

Collision search is done to locate a message that satisfies all sufficient conditions; it involves the use of 2 block messages. The sufficient conditions on the

first block are set in order to control the differentials of the chaining variables on the second block. Since all conditions are conditions of output values, they cannot be satisfied by message modification. Therefore, we don't execute any message modification when searching for a message that satisfies all sufficient conditions of the first block. Fortunately, since the complexity of message search in the first block (2^{14} SHA-0 operations) is much smaller than that of the second block (2^{39} SHA-0 operations), the complexity of the first block does not impact overall complexity. Collision search on the second block is done by using message modification. Furthermore, the early stopping technique can be used to efficiently find a message that satisfies the sufficient conditions. In the early stopping technique, after step 24 is calculated, the sufficient conditions up to step 24 are checked to determine whether they are satisfied or not. If all conditions are satisfied, steps from 25 are calculated. Otherwise, collision search is repeated from the first procedure. It is important to remember that this method still cannot find a message that is assured of satisfying the sufficient conditions in steps 21-24 with probability of almost 1. Submarine modification, proposed in this paper, can satisfy the sufficient conditions in steps 21-24 with probability of almost 1.

Another problem of the existing method is that it is impossible to execute the algorithm proposed by Wang et al. since their description of it is incomplete. We rectify this omission in Appendix B.

4 New Message Modification Techniques

The method of Wang et al. uses message modification to efficiently locate a collision. Their method can efficiently generate messages that satisfying the sufficient conditions up to step 20. However, Wang et al. did not propose message modification for subsequent steps. This section studies message modification, and proposes message modification so as to satisfy the sufficient conditions in steps 21 to 24. In this paper, we call this modification submarine modification. Since the structure of SHA-0 is very similar to those of the MD-family or SHA-1, submarine modification may also be applicable to those hash functions.

4.1 Main Idea of Submarine Modification

Transmission method can be applied to satisfy sufficient conditions for 5 steps from the start step of transmission¹. If we use transmission method to satisfy sufficient condition for after step 22, we need to extend the range where the transmission differential can be started after step 17. Therefore, we use the idea of cancel method in order to extend the range where the transmission differential can be started. If we use the latent differential as the transmission differential, we can extend the range where the transmission differential can be started to step 19 followed by the 5 steps. In the case of SHA-0, the maximum number of latent period after step 17 is $t = 3$ ². As a result, we can satisfy sufficient conditions

¹ We confirm the number of applicable steps by a computer experiment.

² By considering a local collision and message expansion, we can find $t = 3$.

for up to step 24 by combining ideas of cancel method and transmission method. We use the idea of cancel method to create the latent differential for steps 17-19. Since there is no influence for satisfied sufficient conditions in latent period by using cancel method, we can satisfy sufficient conditions for 5 steps from the start step of transmission by applying transmission method. A brief explanation of submarine modification is shown in Figure 1.

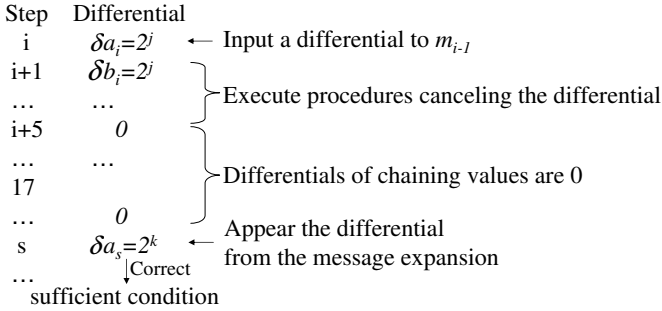


Fig. 1. Outline of Submarine Modification

Remark. In this paper, we apply submarine modification to only the case of steps 21-24. However, submarine modification can be also applied to steps 17-20. We want to note that submarine modification is not limited to only the case of steps 21-24.

4.2 How to Construct Submarine Modification

Submarine modification involves inputting and offsetting differentials and transferring differentials. The procedure of submarine modification is as follows:

1. Decide differentials that satisfy a target sufficient condition in step j ($j \geq 17$) by considering the transfer of differentials. (The idea of transmission method)
2. Decide the method for inputting and offsetting differentials before step 16 to yield the necessary differentials in step j . (The idea of cancel method)

4.3 Proposal of Submarine Modification

There are 4 sufficient conditions from steps 21 to 24: $a_{21,4} = a_{20,4}$ (or $a_{21,4} \neq a_{20,4}$), $a_{22,2} = m_{21,2}$, $a_{22,4} = a_{21,4}$ (or $a_{22,4} \neq a_{21,4}$), $a_{23,2} = m_{22,2}$. In this section, we propose message modification to satisfy each of these sufficient conditions.

Theorem 1. *Suppose we set following conditions as Extra Conditions. $a_{6,6} = m_{5,6}$, $m_{6,11} \neq m_{5,6}$, $m_{7,6} = m_{5,6}$, $a_{7,4} = 0$, $a_{8,4} = 1$, $m_{10,4} \neq m_{5,6}$. If we modify the message as shown below, the sufficient condition $a_{21,4} = a_{20,4}$ (or $a_{21,4} \neq a_{20,4}$) is satisfied with probability of almost 1.*

$$m_5 \leftarrow m_5 \oplus 2^5, m_6 \leftarrow m_6 \oplus 2^{10}, m_7 \leftarrow m_7 \oplus 2^5, m_{10} \leftarrow m_{10} \oplus 2^3$$

In order to satisfy extra conditions, we generate messages that satisfy these extra conditions in advance by a method similar to that used to satisfy the sufficient conditions.

Proof. We explain the change in each chaining variable Theorem 1 is executed in every step.

Step 6. In this step, differential $\delta m_5 = \pm 2^5$ is input. Here, δx is the differential created by message modification on chaining variable x . In this step, a_6 is calculated as follows:

$$a_6 = (a_5 \lll 5) + f(b_5, c_5, d_5) + e_5 + m_5 + k_5.$$

After this equation is calculated, δa_6 becomes $\delta a_6 = \pm 2^5$ because $\delta m_5 = \pm 2^5$. Since $a_{6,6} = m_{5,6}$ is set as the extra condition, $\delta a_6 = \pm 2^5$ does not trigger differential carry. By this condition, since $\delta m_5 = \pm 2^5$ does not cause carry in m_5 , and the sign of δa_6 and $\delta m_{5,6}$ are the same, which confirms that no carry occurs.

Step 7. In step 7, a_7 is calculated as follows:

$$a_7 = (a_6 \lll 5) + f(b_6, c_6, d_6) + e_6 + m_6 + k_6.$$

To ensure $\delta a_7 = 0$, we cancel $\delta a_6 = \pm 2^5$ by $\delta m_6 = \pm 2^{10}$. Since $m_{6,11} \neq m_{5,6}$ was set as the extra condition, the sign of $\delta a_6 = \pm 2^5$ and the sign of $\delta m_6 = \pm 2^{10}$ become opposite, and they cancel each other. Due to this condition, in the case of $m_{5,6} = 0$, $m_{6,11}$ becomes $m_{6,11} = 1$. In this situation, $m_{5,6}$ changes from 0 to 1 because of the differential, and $m_{6,11}$ changes from 1 to 0. Since we have ensured that no carry occurs, δm_5 and δm_6 become $\delta m_5 = 2^5$ and $\delta m_6 = -2^{10}$, respectively. Since $\delta m_5 = 2^5$, δa_6 becomes $\delta a_6 = 2^5$. Therefore, $\delta a_7 = 0$ from $\delta a_6 = 2^5 \lll 5 = 2^{10}$ and $\delta m_6 = -2^{10}$. In the case of $m_{5,6} = 0$ and $m_{6,11} = 1$, a similar analysis finds that δa_7 is assured of being 0.

Step 8. In step 8, a_8 is calculated as follows:

$$a_8 = (a_7 \lll 5) + f(b_7, c_7, d_7) + e_7 + m_7 + k_7.$$

To ensure $\delta a_8 = 0$, we cancel $\delta b_7 = \pm 2^5$ by $\delta m_7 = \pm 2^5$. Since $m_{7,6} = m_{5,6}$ was set as the extra condition, $m_{7,6} = 0$ when $m_{5,6} = 0$. In this situation, $m_{5,6}$ changes from 0 to 1, and $m_{7,6}$ changes from 0 to 1. Since we have ensured that no carry occurs, δm_5 and δm_7 become $\delta m_5 = 2^5$ and $\delta m_7 = 2^5$. Since $\delta m_5 = 2^5$, $\delta a_6 = 2^5$, that is, $\delta b_7 = 2^5$, respectively. Since function f is $f(b_7, c_7, d_7) = (b_7 \wedge c_7) \vee (\neg b_7 \wedge d_7)$, and $c_{7,6} = 0, d_{7,8} = 1$ are ensured to be satisfied by the sufficient conditions; the 2nd bit of $f(b_7, c_7, d_7)$ before differential input is 1, and the 2nd bit of $f(b_7, c_7, d_7)$ after differential input is 0. Therefore, $\delta f(b_7, c_7, d_7)$ becomes -2^5 and is canceled by $\delta m_7 = 2^5$. As a result, δa_8 becomes $\delta a_8 = 0$. In the case of $m_{7,6} = 1$ and $m_{5,6} = 1$, a similar analysis confirms that δa_8 is assured of being 0.

Step 9. In step 9, a_9 is calculated as follows:

$$a_9 = (a_8 \lll 5) + f(b_8, c_8, d_8) + e_8 + m_8 + k_8.$$

Since $a_{7,4} = 0$ is set as the extra condition, we can cancel $\delta c_8 = \pm 2^3$ from the property of function f . Since the function f is $f(b_8, c_8, d_8) = (b_8 \wedge c_8) \vee (\neg b_8 \wedge d_8)$, if $b_{8,4} = 0$, the 4-th bit of $f(b_8, c_8, d_8)$ is equal to $d_{8,4}$, and if $b_{8,4} = 1$, the 4-th bit of $f(b_8, c_8, d_8)$ is equal to $c_{8,4}$. Therefore, since $\delta c_8 = \pm 2^3$, $\delta c_8 = \pm 2^3$ is canceled by setting the extra condition $a_{7,4} = 0$, that is, $b_{8,4} = 0$. As a result, δa_9 becomes 0.

Step 10. In step 10, a_{10} is calculated as follows:

$$a_{10} = (a_9 \lll 5) + f(b_9, c_9, d_9) + e_9 + m_9 + k_9.$$

Since $a_{8,4} = 1$ is set as the extra condition, we can cancel $\delta d_9 = \pm 2^3$ from the property of function f . This basically follows Step 9.

Step 11. In step 11, a_{11} is calculated as follows:

$$a_{11} = (a_{10} \lll 5) + f(b_{10}, c_{10}, d_{10}) + e_{10} + m_{10} + k_{10}.$$

To ensure $\delta a_{11} = 0$, we cancel $\delta e_{10} = \pm 2^3$ by $\delta m_{10} = \pm 2^3$. Since $m_{10,4} \neq m_{5,6}$ is set as the extra condition, $m_{10,4}$ becomes $m_{10,4} = 1$ when $m_{5,6} = 0$. In this situation, $m_{5,6}$ changes from 0 to 1, and $m_{10,4}$ changes from 1 to 0. Since we have ensured that no carry is triggered by the differential, δm_5 and δm_{10} become $\delta m_5 = 2^5$ and $\delta m_{10} = -2^3$, respectively. Since $\delta m_5 = 2^5$, δa_6 becomes $\delta a_6 = 2^5$, that is, $\delta e_{10} = 2^3$. Therefore, $\delta e_{10} = 2^3$ is canceled by $\delta m_{10} = -2^3$, and δa_{11} becomes 0. In the case of $m_{5,6} = 1$ and $m_{10,4} = 0$, a similar analysis shows that δa_{11} becomes 0.

From Step 17. Because of input differentials and message expansion, the following message differentials appear from step 19: $\delta m_{18} = \pm 2^3$, $\delta m_{19} = \pm 2^5$ and $\delta m_{20} = \pm 2^{10}$. $\delta m_{18} = \pm 2^3$ is transferred as shown below, and $a_{21,4} = a_{20,4}$ (or $a_{21,4} \neq a_{20,4}$) is satisfied by $\delta a_{21} = \pm 2^3$.

$$\delta m_{18} = \pm 2^3 \rightarrow \delta a_{19} = \pm 2^3 \rightarrow \delta b_{20} = \pm 2^3 \rightarrow \delta a_{21} = \pm 2^3 \quad \square$$

Remark. We experimentally confirmed that the probability that this message modification can satisfy the target condition without affecting other sufficient conditions is almost 100%. The complexity of this message modification is less than the operations of 2 steps.

Theorem 2. *Suppose we set following conditions as Extra Conditions: $a_{11,21} = m_{10,21}, m_{11,26} \neq m_{10,21}, a_{10,23} = a_{9,23}, a_{12,19} = 0, a_{13,19} = 1, m_{15,19} \neq m_{10,21}, m_{19,26} \neq m_{18,21}$. If we modify a message as shown below, the sufficient condition $a_{22,2} = m_{21,2}$ is satisfied with probability of almost 1.*

$$m_{10} \leftarrow m_{10} \oplus 2^{20}, m_{11} \leftarrow m_{11} \oplus 2^{25}, m_{15} \leftarrow m_{15} \oplus 2^{18}$$

Proof. Since the proof of Theorem 2 is almost the same as the proof of Theorem 1 and due to lack of space, we omit the explanation of this proof.

Remark. We experimentally confirmed that the probability that this message modification can satisfy the target condition without affecting the other sufficient conditions is 97.5%. The complexity of this message modification is less than the operations of 3 steps.

Theorem 3. *Suppose we set the following conditions as Extra Conditions: $a_{11,8} = m_{10,8}, m_{11,13} \neq m_{10,8}, a_{10,10} = a_{9,10}, a_{12,6} = 0, a_{13,6} = 1, m_{15,6} \neq m_{10,8}, m_{19,13} \neq m_{18,8}$. If we modify the message as shown below, sufficient condition $a_{22,4} = a_{21,4}$ (or $a_{22,4} \neq a_{21,4}$) is satisfied with probability of almost 1.*

$$m_{10} \leftarrow m_{10} \oplus 2^7, m_{11} \leftarrow m_{11} \oplus 2^{12}, m_{15} \leftarrow m_{15} \oplus 2^5$$

Proof. Since the proof of Theorem 3 is almost same as that of Theorem 1 and due to lack of space, we omit the explanation of this proof.

Remark. We experimentally confirmed that the probability that this message modification can satisfy the target condition without affecting the other sufficient conditions is almost 100%. The complexity of this message modification is less than the operations of 3 steps.

Theorem 4. *Suppose we set following conditions as Extra Conditions: $a_{11,16} = m_{10,16}, m_{11,21} \neq m_{10,16}, m_{12,16} \neq m_{10,16}, a_{12,14} = 0, a_{13,14} = 1, m_{15,14} \neq m_{10,16}, m_{19,21} \neq m_{18,16}$. If we modify the message as shown below, the sufficient condition $a_{23,2} = m_{22,2}$ is satisfied with probability of almost 1.*

$$m_{10} \leftarrow m_{10} \oplus 2^{15}, m_{11} \leftarrow m_{11} \oplus 2^{20}, m_{12} \leftarrow m_{12} \oplus 2^{15}, m_{15} \leftarrow m_{15} \oplus 2^{13}$$

Proof. Since the proof of Theorem 4 is almost the same as the proof of Theorem 1 and due to lack of space, we omit the explanation of this proof.

Remark. We experimentally confirmed that the probability that this message modification can satisfy the target condition without affecting the other sufficient conditions is 97%. The complexity of this message modification is less than the operations of 4 steps.

4.4 Application to SHA-1

Since a collision attack on SHA-1 [15] is similar to an attack on SHA-0, submarine modification would be applicable to SHA-1. This section considers the application of submarine modification to SHA-1.

Collision search of SHA-1 is done by using message modification as well as collision search of SHA-0. In SHA-1, only message modification for sufficient conditions up to step 22 has been proposed. Therefore, we discuss the possibility of applying submarine modification to realizing the sufficient conditions after step 22 of SHA-1. For example, we discuss message modification to satisfy $a_{23,2} = m_{22,2}$.

Example. Suppose we set following conditions as Extra Conditions: $a_{11,15} = m_{10,15}, m_{11,20} \neq m_{10,15}, a_{10,17} \neq m_{9,17}, a_{12,13} = 0, a_{13,13} = 1, m_{15,13} \neq m_{10,15}, m_{19,21} \neq m_{18,16}$ If we modify the message as shown below, the sufficient condition $a_{23,2} = m_{22,2}$ is satisfied with probability of almost 1.

$$m_{10} \leftarrow m_{10} \oplus 2^{14}, m_{11} \leftarrow m_{11} \oplus 2^{19}, m_{15} \leftarrow m_{15} \oplus 2^{12}$$

However, this message modification can impact other sufficient conditions. An analysis of this is a future work.

If we execute this procedure, the following message differentials appear from step 19 due to message expansion: $\delta m_{18} = \pm 2^{13} \pm 2^{15}, \delta m_{19} = \pm 2^{20}, \delta m_{20} = \pm 2^{15}, \delta m_{21} = \pm 2^{14} \pm 2^{16}, \delta m_{22} = \pm 2^{21}$ Since $m_{19,21} \neq m_{18,16}$ is set as the extra condition, we can minimize the probability of breaking the other sufficient conditions. We omit this explanation since it basically follows that of Theorem 2.

$\delta m_{18} = \pm 2^{13}$ is transferred as shown below, and $a_{23,2} = m_{22,2}$ is satisfied by $\delta a_{23} = \pm 2$.

$$\delta m_{18} = \pm 2^{13} \rightarrow \delta a_{19} = \pm 2^{13} \rightarrow \delta a_{20} = \pm 2^{18} \rightarrow \delta a_{21} = \pm 2^{23} \rightarrow \delta a_{22} = \pm 2^{28} \rightarrow \delta a_{23} = \pm 2$$

Remark. Wang et al. announced an improved version of their original attack on SHA-1 [15] at NIST HASH WORKSHOP 2005 and CT-RSA'06 [12,13].

5 Lack of Sufficient Conditions

When we use the sufficient conditions given by Wang et al. [14], a collision attack does not necessarily succeed even if all sufficient conditions are satisfied. This problem occurs because their approach lacks two conditions. Our analysis, detailed below, showed that the missing conditions are $b_{0,9} = 0$ and $b_{0,11} = 1$.

a_3 is calculated as follows:

$$a_3 = (a_2 \lll 5) + f(b_2, c_2, d_2) + e_2 + m_2 + k_2.$$

We transform the above equation for f .

$$f(b_2, c_2, d_2) = a_3 - (a_2 \lll 5) - e_2 - m_2 - k_2$$

Since $\Delta a_3 = 2 - 2^9 - 2^{11} + 2^{16}, \Delta a_2 = -2^4 - 2^6 + 2^{11}, \Delta e_2 = 0$ and $\Delta m_2 = 2 + 2^6 \pm 2^{31}, \Delta f(b_2, c_2, d_2)$ is calculated as follows:

$$\begin{aligned} \Delta f(b_2, c_2, d_2) &= \Delta a_3 - (\Delta a_2 \lll 5) - \Delta e_2 - \Delta m_2 \\ &= (2 - 2^9 - 2^{11} + 2^{16}) - ((-2^4 - 2^6 + 2^{11}) \lll 5) - 0 - (2 + 2^6 + 2 \pm 31) \\ &= -2^6 \pm 2^{31}. \end{aligned}$$

Since $\Delta b_2 = -2 + 2^6 + 2^{11}, b_{2,2}$ is fixed to change from 1 to 0 due to the differential -2, $b_{2,7}$ is fixed to change from 1 to 0, $b_{2,8}$ is fixed to change from 1 to 0, $b_{2,9}$ is fixed to change from 0 to 1 due to the use of differential 2^6 . The sign of the

change by differential $\pm 2^{31}$ does not have to be considered since it is MSB. Here, we focus on the 7th and 9th bits.

First, we discuss the 7th bit. Wang et al. takes advantage of the fact that $b_{2,7}$ changes from 1 to 0 in order to make differential -2^6 on $f(b_2, c_2, d_2)$. From the property of $f(b_2, c_2, d_2) = (b_2 \wedge c_2) \vee (\neg b_2 \wedge d_2)$, if we set $c_{2,7} = 1$ and $d_{2,7} = 0$, that is, $a_{0,9} = 1$ and $b_{0,9} = 0$ as sufficient conditions, we can make differential -2^6 . However, $b_{0,9} = 0$ was not one of the sufficient conditions described by Wang et al.

We turn now to the 9th bit. $b_{2,9}$ changes from 0 to 1. Wang et al. cancel this influence in function f . From the property of $f(b_2, c_2, d_2) = (b_2 \wedge c_2) \vee (\neg b_2 \wedge d_2)$, if we set $c_{2,9} = d_{2,9}$, that is, $a_{0,11} = b_{0,11}$, we can cancel the influence of the change of $b_{2,9}$. Since $a_{0,11} = 1$ is one of the sufficient conditions given by Wang et al, we need to set $b_{0,11} = 1$ as a sufficient condition. This sufficient condition was not specified by Wang et al.

From the above, we need to use $b_{0,9} = 0$ and $b_{0,11} = 1$ as sufficient conditions in addition to those given by Wang et al.

6 Complexity of Collision Search

Without the additional sufficient conditions the generation of a message that yields a collision will fail with probability $\frac{3}{4}$.

Combining the two additional sufficient conditions with those of Wang et al. and using submarine modification reduces the complexity of collision search to 2^{36} SHA-0 operations. This calculation is given below.

1st block and Step 1-13 of 2nd block. The complexity of generating messages for these steps is insignificant. Refer to the paper written by Wang et al. [14].

Step 14-20 of 2nd block. The complexity of generating messages that satisfy all sufficient conditions in steps 14-20, including message modification, is less than 8 steps.

Step 21 of 2nd block. The complexity of generating messages that satisfy all sufficient conditions up to step 21 including submarine modification is less than,

$$8 + 1 + \frac{1}{2} \cdot 2 = 10.$$

Step 22 of 2nd block. The complexity of generating messages that satisfy all sufficient conditions up to step 22 including submarine modification is calculated as follows: Let the complexity where conditions up to step 22 are satisfied and the number of times m_{14}, m_{15} is chosen is less than i times $x_{22,i}$. In this situation, the following equation below is valid. Here $x_{22,0} = 0$.

$$x_{22,i} = \left(\frac{1}{2} \cdot 0.025\right)^{i-1} \cdot \left(10 + 1 + \frac{1}{2} \cdot 3 + \frac{1}{2} \cdot 3\right) + x_{22,i-1}$$

The complexity is about 15 steps since $\lim_{i \rightarrow \infty} x_{22,i} \approx 15$.

Table 2. An example of generated collision pair

M_{1block}	f459644c b87cdae1 ed98d4a6 7f5c304b a8606648 073dda8d 9f044c3a 2386c95f 8b611aa4 d66ed3b9 c4854f6e d57662b3 d687ebe0 f61cefe5 6d0252c2 01f298bc
h_{1block}	41f3e784 96831ef3 563e0aa9 d7def7ba 232e8581
M_{2block}	76c21fb3 8a725c5a 13a6039c a23c1950 53e65762 b70bbb88 705ec5b6 079e5dd5 f58793f6 d67d305e 352ee1b8 87c36500 fd012cb5 a51c4269 6a72aabd 7a2449cc
M'_{2block}	f6c21ff1 8a725c5a 93a603de a23c1910 53e65722 b70bbbca f05ec5b4 879e5dd7 f58793b6 567d305e b52ee1f8 07c36502 fd012cb7 251c4229 ea72aabd fa24498c
h_{2block}	cad681a1 354105dc ac31607b 6ccaba44 c76d1948

Step 23 of 2nd block. The complexity of generating messages that satisfy all sufficient conditions up to step 23 including submarine modification is calculated as follows: Let the complexity where conditions up to step 23 are satisfied and the number of times m_{14}, m_{15} is chosen is less than i times $x_{23,i}$. In this situation, the following equation below is valid. Here $x_{23,0} = 0$.

$$x_{23,i} = \left(\frac{1}{2} \cdot 0.03\right)^{i-1} \cdot \left(15 + 1 + \frac{1}{2} \cdot 4\right) + x_{23,i-1}$$

The complexity is about 18 steps since $\lim_{i \rightarrow \infty} x_{23,i} \approx 18$.

Step $i(i = 24 - 80)$ of 2nd block. Let the complexity of generating messages that satisfy all sufficient conditions up to the $i - 1$ step be y_{i-1} . If there are n_i sufficient conditions in the i -th step, the probability that all of them are satisfied is 2^{-n_i} . Therefore, y_i , the complexity of generating messages that satisfy all sufficient conditions up to the i -th step, is $y_i = (y_{i-1} + 1) \cdot 2^{n_i}$. From this equation, $y_{80} = 6180766429108$. This is equivalent to 2^{36} SHA-0 operations. From the above consideration, the total complexity of collision search is 2^{36} SHA-0 operations.

Remark. There is a possibility the collision attack could be further improved by using another differential path. We discuss this topic in Appendix A.

7 Conclusion

In this paper, we proposed submarine modification, message modification that can satisfy the sufficient conditions in steps 21-24. Moreover, we showed that submarine modification is applicable to SHA-1. We also showed that the sufficient conditions given by Wang et al. are incomplete since they are missing $b_{0,9} = 0$ and $b_{0,11} = 1$. Therefore, even if a message that satisfies all sufficient conditions given by Wang et al. is discovered, a collision generation may fail with probability $\frac{3}{4}$. By utilizing the two additional sufficient conditions and submarine modification, the complexity of collision search is reduced to 2^{36} SHA-0 operations.

Table 2 shows a collision found by using the technique proposed herein. M_{1block} is a message of the 1st block, h_{1block} is the output of the compression function of the 1st block. M_{2block} is a message for the 2nd block, M'_{2block} is

a message of 2nd block after the differential is input, h_{2block} is the output of the compression function of 2nd block.

Acknowledgement. We would like to thank The Telecommunications Advancement Foundation for supporting our research.

References

1. Eli Biham and Rafi Chen. *Near Collisions of SHA-0*. CRYPTO'04, LNCS 3152, pp290–305, Springer-Verlag, 2004.
2. Eli Biham, Rafi Chen, A. Joux, P. Carribault, W. Jalby and C. Lemuet. *Collisions in SHA-0 and Reduced SHA-1*. EUROCRYPT'05, LNCS 3494, pp36–57, Springer-Verlag, 2005.
3. Florent Chabaud and Antoine Joux. *Differential Collisions in SHA-0*. CRYPTO'98, LNCS 1462, pp56–71, Springer-Verlag, 1998.
4. Antoine Joux. *Collision for SHA-0*. Runm session of CRYPTO2004, August 2004.
5. NIST. *Secure hash standard*. Federal Information Processing Stacdard, FIPS-180, May 1993.
6. NIST. *Secure hash standard*. Federal Information Processing Stacdard, FIPS-180-1, April 1995.
7. Xiaoyun Wang, Dengguo Feng, Hui Chen, Xuejia Lai and Xiuyuan Yu. *Collision for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*. In Rump Session of CRYPTO'04 and Cryptology ePrint Archive, Report 2004/199.
8. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen and Xiuyuan Yu. *Cryptanalysis of the Hash Functions MD4 and RIPEMD*. EUROCRYPT'05, LNCS 3494, pp1–18, Springer-Verlag, 2005.
9. Xiaoyun Wang and Hongbo Yu. *How to Break MD5 and Other Hash Functions*. EUROCRYPT'05, LNCS 3494, pp19–35, Springer-Verlag, 2005.
10. Xiaoyun Wang. *The Collision Attack on SHA-0*. In Chinese, to appear on www.infosec.edu.cn, 1997.
11. Xiaoyun Wang. *The Improved Collision Attack on SHA-0*. In Chinese, to appear on www.infosec.edu.cn, 1998.
12. Xiaoyun Wang, Andrew C Yao, and Frances Yao. *Cryptanalysis on SHA-1 Hash Function*. Keynote Speech at CRYPTOGRAPHIC HASH WORKSHOP.
13. Xiaoyun Wang. *Cryptanalysis of Hash functions and Potential Dangers*. Invited Talk at CT-RSA 2006.
14. Xiaoyun Wang, Hongbo Yu and Yiqun Lisa Yin. *Efficient Collision Search Attack on SHA-0*. CRYPTO'05, LNCS 3621, pp1–16, Springer-Verlag, 2005.
15. Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu. *Finding Collisions in the Full SHA-1*. CRYPTO'05, LNCS 3621, pp17–36, Springer-Verlag, 2005.

A A Study of Other Disturbance Vectors

Wang et al. chose a disturbance vector under the condition that the sufficient conditions up to step 20 can be satisfied by message modification. Therefore, they chose a disturbance vector to minimize the number of sufficient conditions after step 20. However, submarine modification can satisfy sufficient conditions

up to step 24 can be satisfied by message modification. Therefore, we expect that if we choose a disturbance vector to minimize the number of sufficient conditions after step 24, we can generate a collision with complexity under 2^{36} SHA-0 operations. If we use the disturbance vector chosen by Wang et al, the number of conditions after step 24 is 38. However, by using the disturbance vector shown in Table 3, the number of conditions after step 24 is 37. Therefore, we expect that the disturbance vector shown in Table 3 enables us to generate a collision with complexity under 2^{36} SHA-0 operations. Additional analysis on this matter is a future task.

Table 3. A Disturbance Vector for Reduced Complexity

i	value
$-5, \dots, -1$	0 1 1 1 0
$0, \dots, 19$	0 0 0 0 0 1 1 1 0 0 1 1 0 1 1 1 1 1 0 1
$20, \dots, 39$	0 1 1 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0 1 0
$40, \dots, 59$	0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0
$60, \dots, 79$	0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0

B Complement of Collision Search by Wang et al.

B.1 2nd Bit and 7th Bit of Messages

The complexity claims of Wang et al. claim address only the sufficient conditions of chaining variables. They don't consider the complexity of satisfying the sufficient conditions of messages. However, when a random message is generated, it must satisfy the sufficient conditions of messages, and this takes a few steps. This raises the complexity of collision search. This increase can be suppressed by fixing the 2nd bit and 7th bit of the messages in advance in order to ensure satisfaction of the sufficient conditions.

B.2 Sufficient Conditions Given by Wang et al.

The sufficient conditions of Wang et al. include those for $a_{13,4}$, $a_{14,4}$, $a_{15,4}$, $a_{16,4}$, $a_{17,2}$. These values depend on the method used to fix the 2nd and 7th bits of the messages (Discussed in Appendix B.1). That is, if a fixing method different from that of Wang et al. is chosen, the sufficient conditions for $a_{13,4}$, $a_{14,4}$, $a_{15,4}$, $a_{16,4}$, $a_{17,2}$ are also changed.

Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions

Scott Contini¹ and Yiqun Lisa Yin²

¹ Macquarie University, Centre for Advanced Computing – ACAC,
NSW 2109, Australia

scontini@comp.mq.edu.au

² Independent Consultant, Greenwich CT, USA

yiqun@alum.mit.edu

Abstract. In this paper, we analyze the security of HMAC and NMAC, both of which are *hash-based* message authentication codes. We present *distinguishing, forgery, and partial key recovery attacks* on HMAC and NMAC using collisions of MD4, MD5, SHA-0, and reduced SHA-1. Our results demonstrate that the strength of a cryptographic scheme can be greatly weakened by the insecurity of the underlying hash function.

1 Introduction

Many cryptographic schemes use hash functions as a primitive. Various assumptions are made on the underlying hash function in order to prove the security of the scheme. For example, some proofs assume that the hash function behaves as a random oracle, while other proofs only assume collision resistance. With the continuing development in hash function research, especially several popular ones are no longer secure against collision attacks, a natural question is whether these attacks would have any impact on the security of existing *hash-based* cryptographic schemes.

In this paper, we focus our study on HMAC and NMAC, which are hash-based message authentication codes proposed by Bellare, Canetti and Krawczyk [2]. HMAC has been implemented in widely used security protocols including SSL, TLS, SSH, and IPsec. NMAC, although less known in the practical world, is the theoretical foundation of HMAC — existing security proofs [2,1] were first given for NMAC and then extended to HMAC. It is commonly believed that the two schemes have identical security.

The constructions of HMAC and NMAC are based on a *keyed hash function* $F_k(m) = F(k, m)$, in which the IV of F is replaced with a secret key k . NMAC has the following nested structure: $\text{NMAC}_{(k_1, k_2)}(m) = F_{k_1}(F_{k_2}(m))$, where $k = (k_1, k_2)$ is a pair of secret keys. HMAC is similar to NMAC, except that the key pair (k_1, k_2) is derived from a *single* secret key using the hash function. Hence, we can view HMAC as NMAC plus a key derivation function.

The security of HMAC and NMAC was carefully analyzed by its designers [2]. They showed that NMAC is a pseudorandom function family (PRF)

under the two assumptions that (A1) the keyed compression function f_k of the hash function is a PRF, and (A2) the keyed hash function F_k is *weakly collision resistant*¹. The proof for NMAC was then lifted to HMAC by further assuming that (A3) the key derivation function in HMAC is a PRF. The provable security of HMAC, besides its efficiency and elegance, was an important factor for its wide deployment. However, recent collision attacks on hash functions [21,24] imply that assumption (A2) in the original proof no longer holds when considering concrete constructions such as HMAC-MD5 and HMAC-SHA1. To fix this problem, Bellare recently showed [1] that NMAC is a PRF under the sole assumption that the keyed compression function f_k is a PRF. This implies that the security of HMAC now depends only on assumptions (A1) and (A3). The main advantage of the new analysis is that the proof assumptions do not seem to be refuted by existing attacks on hash functions.

The new security proofs are quite satisfying, especially since they are based on relatively weak assumptions of the underlying hash function. On the other hand, they have also raised interesting questions as whether the proof assumptions indeed hold for popular hash functions. In particular, does any existing collision attack on a hash function compromise the PRF assumption? And if so, does it lead to possible attacks on HMAC and NMAC?

1.1 Summary of Main Results

In this paper, we analyze the security of HMAC and NMAC. We answer the aforementioned questions in the affirmative by constructing various attacks on HMAC and NMAC based upon weaknesses of the underlying hash function.

Our analysis is based upon existing analyses of hash functions, especially the attacks on MD4, MD5, SHA-0, and reduced SHA-1 presented in [25,9,10,7]. We first show that the *collision differential path* in these earlier attacks can be used to construct *distinguishing attacks* on the keyed compression function f_k . Hence, for MD4, MD5², SHA-0, and reduced SHA-1, f_k is *not* a PRF.

Building upon the above attacks, we show how to construct distinguishing, forgery, and partial key recovery attacks on HMAC and NMAC when the underlying hash functions are MD4, MD5, SHA-0, and reduced SHA-1. The complexity of our attacks is closely related to the total probability of the collision differential path, and in some cases it is less than the $2^{n/2}$ generic bound for birthday-type attacks. A summary of our main results is given in Table 1. We remark that in our key recovery attack the adversary can retrieve the entire inner key k_2 . This can greatly weaken the security of the scheme. In particular, when the *keyed* inner function is degraded to a hash function with a *known* IV, further attacks such as single-block forgeries become possible.

¹ Please refer to Section 3 for precise definitions of f_k and F_k . The notion of weakly collision resistant (WCR) was introduced in [2]. Roughly, F_k is WCR if it is computationally infeasible to find $m \neq m'$ s.t. $F_k(m) = F_k(m')$ for hidden k .

² In the case of MD5, f_k is not a PRF under *related-key attacks*.

Table 1. Result summary: number of queries in our attacks on HMAC/NMAC

	hash function	distinguish & forgery attacks	key recovery attacks	comments
HMAC/NMAC	MD4	2^{58}	2^{63}	
NMAC	MD5	2^{47}	2^{47}	related-key attacks
HMAC/NMAC	SHA-0	2^{84}	2^{84}	
HMAC/NMAC	reduced SHA-1	2^{34}	2^{34}	inner function is 34 rounds

1.2 Use of Hash Collisions in Our Attacks

Our attacks on HMAC and NMAC are based on collisions of the keyed inner function F_{k_2} . The main reason that an adversary can observe such collisions is that in our scenario the outer function F_{k_1} , although hiding the output of the inner function, does not *hide* the occurrence of an inner collision.

In our key recovery attacks, each bit of collision information – whether or not a collision occurs from a set of properly chosen messages – roughly reveals one bit of the inner key. This is due to the fact that a collision holds information about the entire hash computation, and hence the secret key. Our techniques illustrate that collisions within a hash function can potentially be very dangerous to the security of the upper-layer cryptographic scheme.

1.3 Other Results

General framework for analyzing HMAC and NMAC. We extend the approach in our attacks to provide a general framework for analyzing HMAC and NMAC. This framework also points to possible directions for hash function attacks that most likely lead to further improved attacks on HMAC and NMAC.

Attacks on key derivation in HMAC-MD5. We study the key derivation function in HMAC-MD5, which is essentially the MD5 compression function keyed through the *message input*. We describe distinguishing and second preimage attacks on the function with complexity much less than the theoretical bound.

New modification technique. We develop a new message modification technique in our key recovery analysis. In contrast with Wang’s techniques [21,22], our method does not require full knowledge of the internal hash computation process. We believe that our new technique may have other applications.

1.4 Implications

In practice, HMAC is mostly implemented with MD5 or SHA-1. To a much lesser extent, there is some deployment of HMAC-MD4 (for example, see [12]). We are not aware of any deployment of NMAC. The attacks presented in this paper do not imply any immediate practical threat to implementations of HMAC-MD5 or HMAC-SHA1. However, our attacks on HMAC-MD4 may not be out of range of some adversaries, and therefore it should no longer be used in practice.

We emphasize that our results on HMAC complement, rather than contradict, the analysis in [2,1]. While the designers proved that HMAC is secure under certain assumptions on the underlying hash function, we show that attacks are possible when these assumptions do not hold.

1.5 Organization of the Paper

In Section 3, we provide brief descriptions of HMAC, NMAC and the MDx family. In Section 5, we present all three types of attacks on NMAC-MD5, which is based on the MD5 pseudo-collision (Section 4). The simplicity of the underlying differential path in this case facilitates our explanation, especially the technical details of our key recovery attack. For attacks on HMAC and NMAC using other underlying hash functions, the methods are similar and thus we just focus on what is different in each case in Section 6. In Section 7, we describe a general framework for analyzing HMAC and NMAC.

2 Related Work

Our analysis on HMAC and NMAC is closely related to various attacks on hash functions, especially those in the MDx family. In addition, our work is also related to the rich literature on message authentication codes. Many early heuristic designs for MACs were broken, sometimes in ways that allowed forgery and key recovery [17,18,19]. These early analyses were the driving force behind proposals with formal security proofs, namely HMAC and NMAC [2]. Since their publication, most of the security analysis was provided by the designers. Recently, Coron *et al.* [11] studied the security of HMAC and NMAC in the setting of constructing iterative hash functions. After our submission to Asiacrypt'06, we learned that Kim *et al.* [15] did independent work on distinguishing and forgery attacks on HMAC and NMAC when the underlying functions are MD4, SHA-0, and reduced SHA-1. They did not consider key recovery attacks.

Some of our attacks are in the related-key setting. Related-key attacks were introduced by Biham [5] and Knudsen [14] to analyze block ciphers. A theoretical treatment of related-key attacks was given by Bellare and Kohno [4]. The relevance of related-key cryptanalysis is debated in the cryptographic community. For example, some suggest that the attacks are only practical in poorly implemented protocols. On the other hand, cryptographic primitives that resist such attacks are certainly more robust, and vulnerabilities can sometimes indicate weaknesses in the design. See the introduction to [13] for example settings in which related-key attacks can be applied. We note that the designers of HMAC and NMAC did not consider the related key setting in their security analysis.

3 Preliminaries

3.1 Hash Functions and the MDx Family

A cryptographic hash function is a mathematical transformation that takes an input message of arbitrary length and produces an output of fixed length, called

the *hash value*. Formal treatment of cryptographic hash functions and their properties can be found in [20]. In practice, hash functions are constructed by iterating a *compression function* $f(cv, x)$ which takes fixed length inputs: a chaining variable cv of n bits and a message block x of b bits. The hash function F is defined as follows: First divide the input message m into x_1, x_2, \dots, x_s according to some preprocessing specification, where each x_i is of length b . Then set the first chaining variable cv_0 as the fixed IV, and compute $cv_i = f(cv_{i-1}, x_i)$ for $i = 1, 2, \dots, s$. The final output cv_s of the iteration is the value of F .

The MDx family of hash functions includes MD4, MD5, SHA-0, SHA-1, and others with similar structure. Here we briefly describe the structure of MD5 and omit others. The compression function of MD5 takes a 128-bit chaining variable and a 512-bit message block. The chaining variable is split into four registers (A, B, C, D) , and the message block is split into 16 message words m_0, \dots, m_{15} . The compression function consists of 4 rounds of 16 steps each, for a total of 64 steps. In each step, the registers are updated according to one of the message words. The initial registers (A_0, B_0, C_0, D_0) are set to be some fixed IV. Each step t ($0 \leq t < 64$) has the following general form³:

$$\begin{aligned} X_t &\leftarrow (A_t + \phi(B_t, C_t, D_t) + w_t + K_t) \lll s_t \\ (A_{t+1}, B_{t+1}, C_{t+1}, D_{t+1}) &\leftarrow (D_t, X_t + B_t, B_t, C_t) \end{aligned}$$

In the above equation, ϕ is a round-dependent Boolean function, K_t is a step-dependent constant, and s_t is a step-dependent rotation amount. In each round, all 16 message words are applied in a different order, and so w_t is one of the 16 message words. After the 64 steps, the final output is computed as $(A_{64} + A_0, B_{64} + B_0, C_{64} + C_0, D_{64} + D_0)$.

3.2 Message Authentication Codes, HMAC and NMAC

A message authentication code is a mathematical transformation that takes as inputs a message and a secret key and produces an output called *authentication tag*. The most common attack on MACs is a *forgery attack*, in which the adversary can produce a valid message/tag pair without knowing the secret key. For MACs that are based on *iterative* hash functions, there is a birthday-type forgery attack [17,3] that requires about $2^{n/2}$ MAC queries, where n is the length of the authentication tag.

HMAC and NMAC are both hash-based MACs. Let F be the underlying hash function and f be the compression function. The basic design approach for NMAC is to replace the fixed IV in F with a secret key (aka keyed via the IV). Following the notation in [2], we use $f_k(x) = f(k, x)$ to denote the keyed compression function and $F_k(x) = F(k, x)$ the keyed hash function. Let (k_1, k_2) be a pair of independent keys. The NMAC function, on input message m and secret key (k_1, k_2) , is defined as:

$$\text{NMAC}_{(k_1, k_2)}(m) = F_{k_1}(F_{k_2}(m)).$$

³ We use a slightly different notation from previous work so that there is a unified description for all the steps.

The construction of HMAC was motivated by practical implementation needs. Since NMAC changes the fixed IV in F into a secret key, this requires a modification of existing implementations of the hash function. To avoid this problem, the designers introduced the fixed-IV variant HMAC. Let const_1 and const_2 be two fixed constants. The HMAC function, on input message m and a single secret key k , is defined as:

$$k_1 = f(IV, k \oplus \text{const}_1) \quad (1)$$

$$k_2 = f(IV, k \oplus \text{const}_2) \quad (2)$$

$$\text{HMAC}_k(m) = \text{NMAC}_{(k_1, k_2)}(m).$$

In the above description for HMAC, we can consider Equations (1) and (2) together as a key derivation function KDF which takes a single secret key k and outputs a pair of keys (k_1, k_2) . That is, $(k_1, k_2) = \text{KDF}(k)$. Hence, HMAC is essentially “KDF + NMAC”. We remark that the term “key derivation function” was not used in [2], but this view of the HMAC construction will be quite convenient for our later analysis.

4 Pseudo-collisions of MD5

In [9], den Boer and Bosselaers analyzed the compression function of MD5 and found pseudo-collisions of the form $f(cv, m) = f(cv', m)$, where cv and cv' are two different IVs. Such pseudo-collisions of MD5 are the basis for our related-key attacks on NMAC-MD5. In this section, we discuss some properties of the pseudo-collisions under the framework of differential cryptanalysis.

Differential cryptanalysis was introduced by Biham and Shamir [8] to analyze the security of DES. The idea also applies to the analysis of hash functions. In a hash collision attack, we consider input pairs with an appropriately defined difference and analyze how the differences in the chaining variables evolve during the hash computation. The intermediate differences collectively are called a *differential path*, and its probability is defined to be the probability that the path holds when averaged over all input pairs satisfying the given difference.

For the MD5 pseudo-collisions in [9], the messages are the same and the input difference is only in the chaining variables. The pair of initial chaining variables (cv, cv') as well as all the intermediate values satisfy the following difference:

$$cv \oplus cv' = (80000000 \ 80000000 \ 80000000 \ 80000000) \stackrel{\text{def}}{=} \Delta^{\text{msb}}. \quad (3)$$

Putting in concrete terms, the differences are only in the most significant bit (MSB) of each register A_t, B_t, C_t, D_t . This simple pattern propagates through all 64 steps of MD5. Because of the extra addition operation at the end, the difference disappears, yielding a pseudo-collision.

The differential path requires the following conditions on the IV:

$$\text{MSB}(B_0) = \text{MSB}(C_0) = \text{MSB}(D_0) = b, \quad (4)$$

where $b = 0$ or 1 . Moreover, the MSBs of the intermediate registers are the same for most of the first round. Namely, for $1 \leq t < 15$,

$$\text{MSB}(A_t) = \text{MSB}(B_t) = \text{MSB}(C_t) = \text{MSB}(D_t) = b.$$

The total probability of the differential is 2^{-46} .

5 Related-Key Attacks on NMAC-MD5

In this section, we present distinguishing, forgery, and partial key recovery attacks on NMAC-MD5 in the related-key setting. In this setting, the goal of the adversary is to break the MAC by obtaining input/output pairs of two MAC oracles whose keys are different but with a known relation.

As described in Section 4, the differential path for the MD5 pseudo-collision holds with probability 2^{-46} . Given the path, we can construct a related-key distinguishing attack on the keyed MD5 compression function that requires about 2^{47} queries. This distinguishing attack is the basis for all three types of attacks on NMAC-MD5. Since the distinguishing attacks on the MD5 compression function and on NMAC-MD5 are nearly identical, we omit the details of the former.

Recall that in NMAC, the inner function F_{k_2} is keyed through the IV. Hence, in our related-key attacks, the difference in the *inner key* k_2 is set according to the input IV difference given by Equation (3). More specifically, we have the following setting for our related-key attacks on NMAC-MD5:

- There are two oracles $\text{NMAC}_{(k_1, k_2)}$ and $\text{NMAC}_{(k'_1, k'_2)}$. The relation between (k_1, k_2) and (k'_1, k'_2) is set as:

$$k_1 = k'_1 \quad \text{and} \quad k_2 \oplus k'_2 = \Delta^{\text{msb}}. \quad (5)$$

- The adversary queries each oracle on input messages of its choice and is given the corresponding authentication tag.

5.1 Related-Key Distinguishing and Forgery Attacks on NMAC-MD5

We first present a related-key distinguishing attack on NMAC-MD5, based upon the lack of pseudorandomness of the keyed MD5 compression function. In this attack, the adversary is given two oracles (O, O') , which can either be the two NMAC oracles as defined by Equation (5) or oracles for truly random functions. The adversary generates 2^{46} random messages and queries both oracles. If a collision $O(m) = O'(m)$ is observed for any message m , it identifies the oracles as NMAC; otherwise, it identifies them as a truly random function.

The correctness of the attack is easy to see: After 2^{46} messages, a collision of the inner function is expected. That is, $F_{k_2}(m) = F_{k'_2}(m)$. Since the outer key k_1 is the same, the inner collision yields a collision for the two NMAC oracles. The complexity is 2^{46} random queries to each oracle, for a total of 2^{47} queries. The

attack succeeds if k_2 satisfies the condition given by Equation (4). Hence, for two random NMAC key pairs which satisfy the relation given by Equation (5), the success probability of our distinguishing attack is $1/4$.

It is worth noticing that the outer function in NMAC, although making the output of the inner function hidden, does not *hide* the occurrence of an inner collision. This property is very useful for converting the distinguishing attack on the inner function (which is the keyed MD5 compression) to a distinguishing attack on NMAC. Such a conversion also applies to HMAC.

The attack can be extended to a forgery attack as follows [17,3]: Once a message m is found that causes a collision of the two NMAC oracles, the adversary queries the *first* oracle on $m||e$ for any extension e and obtains $tag = \text{NMAC}_{(k_1, k_2)}(m||e)$. Then, it produces $(m||e, tag)$ as a forgery for the *second* oracle. Since $\text{NMAC}_{(k_1, k_2)}(m||e) = \text{NMAC}_{(k'_1, k'_2)}(m||e)$, the forged authentication tag is valid. The complexity is 2^{47} random queries plus one *chosen* query. Hence, the total number of queries is about 2^{47} and the success probability is $1/4$.

5.2 Related-Key Key Recovery Attack on NMAC-MD5

We present a partial key recovery attack on NMAC-MD5, in which the adversary can retrieve the entire inner key k_2 in NMAC. This is the most technical part of the paper, so we start with a high level description of the key recovery algorithm consisting of four phases:

- **Phase 1.** The attacker generates random messages until it obtains a message m that causes a collision of the two NMAC oracles.
- **Phase 2.** The attacker modifies certain bits of m to create new messages m^* and observes whether any m^* causes a new collision. This collision information allows the attacker to recover many bits in the intermediate registers $S = (A_{14}, B_{14}, C_{14}, D_{14})$ in the computation of $F_{k_2}(m)$.
- **Phase 3.** Similar to Phase 2, the attacker recovers a few additional bits from other registers, and uses this information to determine more bits of S with a possible small additive error.
- **Phase 4.** The attacker guesses all remaining unknown bits of S and steps through the MD5 computation backwards to get (A_0, B_0, C_0, D_0) – a candidate for k_2 . It verifies whether $F_{k_2}(m) = F_{k'_2}(m)$. If so, it outputs k_2 as the inner key; Otherwise, go back to Phase 1.

Phase 1 and Phase 4 of the key recovery algorithm are fairly straightforward, and so for the rest of the section we focus on Phase 2 and Phase 3. We first explain the main idea and then present detailed analysis.

Main idea. For Phase 2 and Phase 3, the objective is to recover bits of some intermediate registers through collision information. To achieve this goal, we take a closer look at the collision differential paths and analyze what information can be derived from such paths. Let DP_m denote the differential path induced by m , i.e., all the intermediate differences in the computation of $F_{k_2}(m)$ and $F_{k'_2}(m)$.

Since m yields a collision, we know that DP_m follows the differential path for the MD5 pseudo-collision. In particular, for the computation of $F_{k_2}(m)$, we have $\text{MSB}(B_t) = b$ for $1 \leq t < 15$. WLOG, we assume $b = 0$.

For a given step t in the first round, we introduce a new message m^* that is defined based on message m as follows:

$$m_j^* = \begin{cases} m_j & \text{if } 0 \leq j < t \\ m_j + \Delta & \text{if } j = t \\ \text{random} & \text{if } t < j < 16 \end{cases} \quad (6)$$

We next consider the differential path DP_{m^*} , induced by m^* . Since m and m^* are the same up to Step $t-1$, the two paths DP_m and DP_{m^*} are the same until this step. For Step t , let B_{t+1}^* be the newly computed register by replacing m_t with $m_t^* = m_t + \Delta$. We know that B_{t+1}^* will be different from B_{t+1} . A key observation is that if $\text{MSB}(B_{t+1}^*)$ changes from 0 to 1, then the path DP_{m^*} will *drift away* from the collision differential path, and hence the chance of it producing a collision after 64 steps is negligible. More precisely, we have the following lemma.

Lemma 1. *Let m^* be a message defined as in Equation (6), and let p^* be the probability that m^* causes a collision $F_{k_2}(m^*) = F_{k_2'}(m^*)$. If $\text{MSB}(B_{t+1}^*) = 0$, then $p^* = 2^{t-45}$ when averaged over all random m_j^* ($j > t$). If $\text{MSB}(B_{t+1}^*) = 1$, then $p^* \approx 2^{-128}$.*

For a given value Δ , Lemma 1 can be used to detect the MSB of B_{t+1}^* as follows: generate about 2^{45-t} messages satisfying Equation (6) and query both NMAC oracles on these messages. If a collision is observed, then the MSB of B_{t+1}^* is 0; otherwise, the bit is 1.

In what follows, we show how to use the above collision information to recover B_{t+1} . To better illustrate the intuition, we consider a simplified step function where the rotate is *eliminated*. Hence Step t becomes $B_{t+1} = m_t + T$ and $B_{t+1}^* = m_t^* + T$, where the value T has been determined before Step t . To detect bit i of B_{t+1} , we set $m_t^* = m_t + 2^i$. This implies that

$$B_{t+1}^* = B_{t+1} + 2^i. \quad (7)$$

We consider the effect of the above increment, depending on whether bit i of B_{t+1} is 0 or 1:

- If bit i of B_{t+1} is 0, then the increment will not cause a carry. In this case, $\text{MSB}(B_{t+1}^*) = \text{MSB}(B_{t+1}) = 0$, and we will observe a collision in the expected number of queries.
- If bit i of B_{t+1} is 1, then the increment causes a carry. Furthermore, if we can set bits $[(i+1)..30]$ of B_{t+1}^* to be all 1, then the carry will go all the way to the MSB of B_{t+1}^* . In this case, $\text{MSB}(B_{t+1}^*) = \text{MSB}(B_{t+1}) + 1 = 1$, and we will *not* observe a collision.

To ensure carry propagates to the MSB, we set $m_t^* = m_t + 2^i + d$, for an appropriate choice of d . So Equation (7) becomes $B_{t+1}^* = B_{t+1} + 2^i + d$.

The above analysis yields an algorithm for determining B_{t+1} one bit at a time, from bit 30 to bit 0. (Note that we already know bit 31 of B_{t+1} is 0 by assumption.) We refer to this algorithm as the *bit flipping algorithm*, and the complete description is given in Appendix A.

Detailed analysis. The main idea described above generally applies to any register B_t for $0 \leq t < 15$. In Phase 2, the registers to be recovered are

$$(B_{11}, B_{12}, B_{13}, B_{14}) = (A_{14}, D_{14}, C_{14}, B_{14}).$$

The reason why we choose later registers rather than earlier ones is to minimize the number of oracle queries, which is 2^{45-t} per oracle per bit computed of register B_{t+1} . We leave B_{15}, B_{16} free so that there is enough randomness for generating new collisions.

We now consider how to apply the bit flipping algorithm in the presence of rotation. We need to do $B_{t+1}^* = B_{t+1} + 2^i + d$ for $i = 30, 29, \dots, 0$. However, we are not able to do so by just setting $m_t^* = m_t + 2^i + d$ because of the rotation operation \lll_{s_t} . Instead, we use a *modified* bit flipping algorithm (see Appendix A for details). In this algorithm, we set $m_t^* = m_t + 2^{i'} + d'$ where

$$i' + s_t = i \bmod 32 \quad \text{and} \quad d' \lll_{s_t} = d.$$

Note that if addition and rotation could commute, then setting m_t^* as above would have the same effect as $B_{t+1}^* = B_{t+1} + 2^i + d$. Since this is not the case, some error might occur when applying the modified algorithm. Fortunately, the error is manageable — we can show that the modified algorithm almost always succeeds for recovering the most significant $(32 - s_t)$ bits of B_{t+1} . In other words, if it fails, it is almost always on the least significant s_t bits. More precisely, we have the following lemma. The proof is omitted due to space limit.

Lemma 2. *For step t , let p_t be the probability that the modified bit flipping algorithm correctly recovers the most significant $(32 - s_t)$ bits of B_{t+1} , when averaged over all possible input messages m . Then $p_t \geq 1 - 2^{-s_t} - 2^{-s_t-1}$.*

For the four steps $t = 10, 11, 12, 13$, the rotation amounts are $s_t = 17, 22, 7, 12$. Hence, we can use the modified bit flipping algorithm to determine the following bits of the registers:

$$\begin{aligned} A_{14} &= B_{11} : \text{ most significant 15 bits} \\ D_{14} &= B_{12} : \text{ most significant 10 bits} \\ C_{14} &= B_{13} : \text{ most significant 25 bits} \\ B_{14} &= B_{14} : \text{ most significant 20 bits} \end{aligned}$$

In total we already recover 70 bits of the registers. We could proceed to Phase 4 and guess the remaining 58 bits. This would yield a key recovery algorithm with query complexity 2^{47} and time complexity equal to about 2^{58} MD5 operations, which is much less than exhaustive key search.

With refined analysis, we can further reduce the workload by doing an insignificant number of additional queries in Phase 3. We do so by following similar steps as in Phase 2, except recovering bits of earlier registers, namely the most significant $(32 - s_t)$ bits of B_{10}, B_9, B_8 . Once these bits are known, the interaction between successive steps can be used to determine 10 more bits of the registers $(A_{14}, D_{14}, C_{14}, B_{14})$ up to a possible small additive error. Due to space limits, specific details are omitted. Together with an early stopping technique in Phase 4, the remaining workload is at most 2^{45} MD5 operations. This can be reduced further, but 2^{45} is already do-able with moderate computing resources. The total number of queries is still dominated by that of Phase 1, which is 2^{47} .

Implementation results. We have implemented the key recovery attack on NMAC-MD5. In our implementation, we used a reduced-round version of MD5, in which the last round (16 steps) is omitted. Since the attack only depends on properties of the first round, the reduction in rounds does not affect the analysis except that the query complexity is reduced from 2^{47} to 2^{31} . In our experiment, the algorithm correctly recovered the inner key bits.

Remarks on message modification techniques. In the key recovery analysis, we use information about the collision differential paths to derive information about the intermediate registers. To generate useful paths, we developed a new message modification technique that works even when the internal hash computation is unknown due to the presence of the *secret key*.

It is worth comparing our modification techniques with Wang’s original message modification techniques [21,22], which deals with the situation where the entire hash computation is known since there is no secret for a keyless hash function. Note that the objective of the modification is also different for collision attacks and our key recovery attacks: the goal for the former is to modify messages so that collisions can occur with high probability; the goal for the latter is to modify messages so that certain collisions may or may *not* occur, depending upon the value of the secret key.

5.3 Attacks on the KDF in HMAC-MD5

Given our related-key attacks on NMAC-MD5, an immediate question is whether they are applicable to HMAC-MD5. Since the difference between HMAC and NMAC is the extra key derivation function KDF, we analyze properties of KDF in HMAC-MD5, which consists of two functions of the form $k_i = f(IV, k \oplus \text{const}_i)$. Here the MD5 compression function f is used as $f(x, K)$, where $x \in \{0, 1\}^{128}$ and the key $K \in \{0, 1\}^{512}$. For ease of reference, we denote $f(x, K)$ by $g_K(x)$. So $\{g_K\}_{K \in \{0, 1\}^{512}}$ is a family of functions indexed by K .

As noted in Section 5.4 of [1], Rijmen observed that it seems possible to extend the pseudo-collision of MD5 [9] to a distinguishing attack on $\{g_K\}$. Here, we describe the details of such an attack: The adversary generates 2^{46} random pairs (x, x') such that $x \oplus x' = \Delta^{\text{msb}}$, and queries an oracle, which is either g_K or a truly random function. If the adversary observes a collision for any pair, then

it identifies the oracle as g_K ; otherwise, it identifies the oracle as a truly random function. The complexity of the attack is 2^{47} queries.

Recall that the HMAC security proofs [1,2] require KDF to be a PRF. However, the above distinguishing attack implies that the KDF in HMAC-MD5 is *not* a PRF. Despite the non-pseudorandomness, its presence does help HMAC-MD5 to resist our related-key attacks for the following reason. In order to apply the attacks to HMAC-MD5, we would need to set appropriate differences in the single key k and hope that $(k_1, k_2) = \text{KDF}(k)$ would yield the required difference for k_2 while keeping k_1 the same (see Equation (5)). However, this appears to be very difficult, since any differences in k would almost certainly cause differences in both k_1 and k_2 , thus making the attacks impossible.

Of independent interest, we present a *second preimage attack* on g_K , also based on [9]. Here the key K can be either secret or known. The attack works as follows: For a given random input $x \in \{0, 1\}^{128}$, the adversary sets x' such that $x \oplus x' = \Delta^{\text{msb}}$, and outputs x' as a second preimage of x . The success probability is about 2^{-48} , since the probability that x satisfies Equation (4) is 2^{-2} , and the probability that the pair (x, x') then follows the differential path to produce a collision is 2^{-46} (meaning x' is a second preimage of x). Hence, the above attack requires $O(1)$ workload, no queries, and succeeds with probability 2^{-48} , which is much higher than the 2^{-128} theoretical bound.

6 Attacks on HMAC/NMAC with Other Hash Functions

The basis for our attacks on NMAC-MD5 is a collision differential path for the keyed MD5 compression function that holds with *relatively large* probability. The same ideas and techniques also apply to other underlying hash functions such as MD4, SHA-0, and reduced SHA-1. In this section, we present three types of attacks on HMAC and NMAC for these underlying hash functions, all in the *standard* setting.

6.1 Attacks on HMAC/NMAC-MD4

MD4 has long been known to be insecure, but it was an open question whether HMAC-MD4 can still be used as a PRF or a secure MAC. We answer the question in the negative by presenting attacks on HMAC/NMAC-MD4.

Our attacks are based upon the second preimage attack on MD4 by Yu *et al.* [25]. Table 3 of [25] gives a differential path that leads to a collision with probability 2^{-62} . The details that are most relevant to our attacks are the message difference: there is only a one-bit difference in one of the message words, namely, $m_4 \oplus m'_4 = 2^i$, and the path holds for any i ($0 \leq i < 32$), for a total of 32 possible paths. Given the paths, we can mount a distinguishing attack on the keyed MD4 compression function, implying that the function is *not* a PRF.

For our distinguish attack on HMAC-MD4, there is only a *single* oracle O , which can be either HMAC_k or a truly random function. The adversary generates about 2^{62} message pairs (m, m') such that $m_4 \oplus m'_4 = 2^i$ for some i , queries the oracle, and observes whether a collision $O(m) = O(m')$ occurs. If so, it identifies

the oracle as HMAC; otherwise, it identifies it as a truly random function. The expected query complexity is 2^{63} , and the success probability is one. From the collision, a forgery attack easily follows (similar to Section 5.1) which requires an additional chosen query.

We can reduce the query complexity to 2^{58} by using a structure, which is a common trick in differential cryptanalysis. The idea is to take advantage of the multiple differential paths by generating input pairs (m, m') in a more compact way as follows: First, generate 2^{26} random m_3 (it can actually be any message word m_j as long as $j \neq 4$). Second, for each m_3 , generate all 2^{32} possible values for m_4 . Hence, the total number of messages is 2^{58} . It is easy to show that the 2^{58} messages collectively create 2^{62} pairs of (m, m') for which $m_4 \oplus m'_4 = 2^i$ for some i . One of the pairs is expected to produce a collision.

We can construct a partial key recovery attack on HMAC-MD4 following similar phases as that of NMAC-MD5. Given the form of the 32 differential paths and their associated conditions, it is better to use only one path ($i = 22$) for key recovery. Our analysis shows that the query complexity is roughly 2^{63} and the remaining computation is order 2^{40} MD4 operations.

6.2 Attacks on HMAC/NMAC-SHA0

Chabaud and Joux [10] presented the first collision attack on SHA-0 with complexity 2^{61} . Their analysis also introduced important concepts such as local collisions and disturbance vectors, which prove to be the basis for all subsequent attacks on SHA-0 and SHA-1. The differential path used in their attack holds with probability $p = 2^{-83}$ (see Table 4 in [10] for detailed calculation). We can use the differential path to construct distinguish and forgery attack on HMAC-SHA0 with query complexity 2^{84} . One subtle issue for SHA-0 (and SHA-1) is that we should generate message pairs so that they not only satisfy the required message difference but also extra conditions on certain message bits.

A partial key recovery attack on HMAC-SHA0 can also be constructed. In fact, the analysis would be much simpler than that of NMAC-MD5 due to the particular form of the SHA-0 (and SHA-1) step function, which is $A_i = (A_{i-1} \lll 5) + f_i(B_{i-1}, C_{i-1}, D_{i-1}) + E_{i-1} + m_{i-1} + k_i$. Since there is no rotation associated with the message word, we can use the bit flipping algorithm directly (rather than the modified version) to recover the register A_i . Our analysis shows that the query complexity is about 2^{84} , and the time complexity is about 2^{60} .

6.3 Attacks on Reduced-Round Variants of HMAC/NMAC-SHA1

Biham *et al.* [7] presented collision attacks on several reduced-round variants of SHA-1. Their attack on 34-round SHA-1 used a disturbance vector with very low Hamming weight (see Table 1 of [7]). Based on this vector, we calculated the probability of the differential path to be 2^{-33} , and it holds for half of the randomly chosen IVs. This path implies that 34-round SHA-1 is not a PRF. Using our techniques developed earlier, we can construct all three types of attacks on HMAC-SHA1 when the inner function is reduced to 34 rounds. The query complexity is about 2^{34} and the success probability is $1/2$ for a random key.

6.4 Further Improvements

It is possible to further improve the complexity of our attacks. Krawczyk [16] pointed out a useful tradeoff between query complexity and the success probability of the attacks. More specifically, we can construct new attacks with 2^t queries and success probability 2^{t-q} , where 2^q is the number of queries in our original attacks and $1 \leq t \leq q$. Biham [6] suggested that attacks on HMAC can be extended to 40-round SHA-1 using results in [7].

7 A General Framework for Analyzing HMAC/NMAC

In this section we extend the approach in our attacks to provide a general framework for analyzing HMAC/NMAC. Let DP be a collision differential path for the compression function f , and let $\Delta = (\Delta cv, \Delta m)$ be the required input difference for the path. Suppose that the path holds with probability at least $P_0 = 2^{-w}$ for a fraction q of all randomly chosen inputs (cv, cv') and (m, m') satisfying Δ . We consider two cases depending on Δcv :

- $\Delta cv = 0$. In this case, the path DP yields a real collision. The attacks to be considered are in the standard setting and apply to both HMAC and NMAC.
- $\Delta cv \neq 0$. In this case, the path DP yields a pseudo-collision. The attacks to be considered are in the related-key setting and apply only to NMAC.

There are three types of possible attacks, all having success probability q .

1. *Distinguishing attack.* The complexity is about $O(2^{w+1})$ queries.
2. *Forgery attack.* If the hash function F is iterative, the distinguishing attack implies a forgery attack with one additional chosen query.
3. *Key recovery attack.* If F has similar step functions as MDx, the collision path may allow the recovery of the inner key in HMAC and NMAC. The query complexity is $O(2^{w+1})$, and the time complexity depends on the form of the collision path.

To beat the generic birthday-type forgery attack, we need to find a collision differential path such that $P_0 > 2^{-n/2}$, and to beat the exhaustive key search attack, we need $P_0 > 2^{-n}$. Hence, the above general framework reduces the problem of attacking HMAC/NMAC to the problem of finding a “good” collision differential path for the underlying compression function.

Finding suitable differential paths. There have been many collision attacks on hash functions, each relying on a specific differential path. One important point is that a differential path that works best for finding collisions may not be the best for the purpose of attacking HMAC and NMAC. To better explain this, we introduce a variable P_r , which is the probability of the differential path from Step r to the last step.

- For collision attacks, we should select a path such that P_r is minimized, assuming message modification techniques can apply up to Step $r-1$ of the hash function.
- For attacks on HMAC and NMAC, we should select a path such that P_0 is minimized.

For example, for the purpose of analyzing HMAC-SHA0, Chabaud and Joux's attack offers a better differential path than the improved collision attack in [23], since the probability P_0 associated with the differential path in the former attack is much larger than the latter.

To break HMAC-MD5, we would need to find differential paths that hold with large enough probability P_0 and lead to real collisions. The differential path in Wang's MD5 attack [21] was constructed to minimize P_{17} ($\approx 2^{-37}$) so that it works best with modification techniques. The total probability P_0 of the path is only about 2^{-300} . So far, improvements to the MD5 attack were all due to refined modification techniques: nobody has discovered new differential paths. An open question is whether differential paths for MD5 with $P_0 > 2^{-128}$ can be found. New automated search methods may provide promising ways for finding such differential paths.

Acknowledgements. We thank Mihir Bellare and Hugo Krawczyk for valuable suggestions on an early draft of this work. We thank Eli Biham for enlightening discussions. We also thank Lily Chen, Antoine Joux, Josef Pieprzyk, and the Asiacrypt reviewers for helpful comments. The first author was supported by ARC grant DP0663452.

References

1. M. Bellare. *New Proofs for NMAC and HMAC: Security without Collision-Resistance*. ePrint archive, May 2006. To appear in CRYPTO 2006.
2. M. Bellare, R. Canetti and H. Krawczyk. *Keyed Hash Functions for Message Authentication*. CRYPTO 1996.
3. M. Bellare, R. Canetti and H. Krawczyk. *Pseudorandom Functions Revisited: the Cascade Construction*. FOCS 1996.
4. M. Bellare and T. Kohno. *A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications*. EUROCRYPT 2003.
5. E. Biham. *New Types of Cryptanalytic Attacks Using Related Keys*. EUROCRYPT 1993.
6. E. Biham. *Personal communication*. August 2006.
7. E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby and C. Lemuet. *Collisions in SHA-0 and Reduced SHA-1*. EUROCRYPT 2005.
8. E. Biham and A. Shamir. *Differential Cryptanalysis of DES-like Cryptosystems*. CRYPTO 1990.
9. B. den Boer and A. Bosselaers. *Collisions for the Compression Function of MD5*. EUROCRYPT 1993.
10. F. Chabaud and A. Joux. *Differential Collisions in SHA-0*. CRYPTO 1998.
11. J.-S. Coron, Y. Dodis, C. Malinaud and P. Puniya. *Merkle-Damgard Revisited : how to Construct a Hash Function*. CRYPTO 2005.
12. The GNU Crypto project. <http://www.gnu.org/software/gnu-crypto/>.
13. J. Kelsey, B. Schneier, and D. Wagner. *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA*. ICICS 1997.
14. L. Knudsen. *Cryptanalysis of LOKI91*. AusCrypt 1992.
15. J. Kim, A. Biryukov, B. Preneel and S. Lee. *On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1*. SCN 2006. Also available at <http://eprint.iacr.org/2006/187>.

16. H. Krawczyk. *Personal communication*. June 2006.
17. B. Preneel and P.C. van Oorschot. *MDx-MAC and Building Fast MACs from Hash Functions*. CRYPTO 1995.
18. B. Preneel and P.C. van Oorschot. *On the Security of Two MAC Algorithms*. EUROCRYPT 1996.
19. B. Preneel and P. C. van Oorschot. *A key recovery attack on the ANSI X9.19 retail MAC*. Electronics Letters 32(17), 1996.
20. P. Rogaway and T. Shrimpton. *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance*. FSE 2004.
21. X. Wang and H. Yu. *How to Break MD5 and Other Hash Functions*. EUROCRYPT 2005.
22. X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. *Cryptanalysis of the Hash Functions MD4 and RIPEMD*. EUROCRYPT 2005.
23. X. Wang, H. Yu, and Y.L. Yin. *Efficient Collision Search Attacks on SHA-0*. CRYPTO 2005.
24. X. Wang, Y.L. Yin, and H. Yu. *Finding Collisions in the Full SHA-1*. CRYPTO 2005.
25. H. Yu, G. Wang, G. Zhang, and X. Wang. *The Second-Preimage Attack on MD4*. CANS 2005. Available on Springerlink web site.

A The Bit Flipping Algorithms

We first give the bit flipping algorithm in Figure 1. This is for the simplified MD5 step function where the rotation is eliminated.

```

For  $j = 0, \dots, t-1$ , set  $m_j^* = m_j$ 
Set  $d = 0$  (a)
For  $i = 30$  downto 0 do (b)
{
  Set  $m_t^* = m_t + 2^i + d$  (c)
  Repeat order  $2^{46-t}$  times
  {
    Choose  $m_{t+1}^*, \dots, m_{15}^*$  at random.
    /* now all 16 words of  $m^*$  have been set */
    Query the two nmac oracles on  $m^*$ 
    If there is a collision, then
    {
      Bit  $i$  of  $B_{t+1}$  is 0
      Set  $d = d + 2^i$  (d)
      break;
    }
  }
}
If no collision found, then bit  $i$  of  $B_{t+1}$  is 1
}

```

Fig. 1. Bit flipping algorithm for computing B_{t+1}

The modified bit flipping algorithm is similar, except the following four steps:

- Step (a) \Rightarrow **Set** $d' = 0$
- Step (b) \Rightarrow **For** $i' = 30 - s_t$ **downto** 0 **do**
- Step (c) \Rightarrow **Set** $m_t^* = m_t + 2^{i'} + d'$
- Step (d) \Rightarrow **Set** $d' = d' + 2^{i'}$

New Guess-and-Determine Attack on the Self-Shrinking Generator*

Bin Zhang and Dengguo Feng

State Key Laboratory of Information Security,
Institute of Software, Chinese Academy of Sciences,
Beijing 100080, P.R. China
martin_zhangbin@yahoo.com.cn

Abstract. We propose a new type of guess-and-determine attack on the self-shrinking generator (SSG). The inherent flexibility of the new attack enables us to deal with different attack conditions and requirements smoothly. For the SSG with a length L LFSR of arbitrary form, our attack can reliably restore the initial state with time complexity $O(2^{0.556L})$, memory complexity $O(L^2)$ from $O(2^{0.161L})$ -bit keystream for $L \geq 100$ and time complexity $O(2^{0.571L})$, memory complexity $O(L^2)$ from $O(2^{0.194L})$ -bit keystream for $L < 100$. Therefore, our attack is better than all the previously known attacks on the SSG and especially, it compares favorably with the time/memory/data tradeoff attack which typically has time complexity $O(2^{0.5L})$, memory complexity $O(2^{0.5L})$ and data complexity $O(2^{0.25L})$ -bit keystream after a pre-computation phase of complexity $O(2^{0.75L})$. It is well-known that one of the open research problems in stream ciphers specified by the European STORK (Strategic Roadmap for Crypto) project is to find an attack on the self-shrinking generator with complexity lower than that of a generic time/memory/data tradeoff attack. Our result is the best answer to this problem known so far.

Keywords: Stream cipher, Self-shrinking, Guess-and-determine, Linear feedback shift register (LFSR).

1 Introduction

The self-shrinking generator is an elegant keystream generator proposed by W. Meier and O. Staffelbach at EUROCRYPT'94 [22]. It applies the shrinking idea [7] to only one maximal length LFSR and generates the keystream according to the following rule: let $a = a_0, a_1, \dots$ be a binary sequence produced by the LFSR, consider the bit pair (a_i, a_{i+1}) , if $a_i = 1$, output a_{i+1} as a keystream bit, otherwise no output is produced. It is suggested in [22] that the key of the SSG consists of the initial state of the LFSR and (preferably) also of the LFSR

* Supported by the National Natural Science Foundation of China (Grant No. 90604036, 60373047) and the National Grand Fundamental Research 973 program of China (Grant No. 2004CB318004).

feedback logic. As in other articles, e.g. [18,26,31,3], we assume that the primitive feedback polynomial is known to the attacker.

Although many LFSR based stream ciphers are found vulnerable to (fast) correlation attacks [4,5,14,15,16,23,24,25] and algebraic attacks [1,2,8,9], the self-shrinking generator has shown remarkable resistance against such cryptanalysis. For a length L LFSR, the previously known best concrete attack is the BDD attack in [18], which has time complexity $O(2^{0.656L})$ at the expense of $O(2^{0.656L})$ memory from $\lceil 2.41 \cdot L \rceil$ bits keystream. One of the open research problems in stream ciphers specified by the STORK (Strategic Roadmap for Crypto) project [29] is to find an attack on the self-shrinking generator with complexity lower than that of a generic time/memory/data (TMD) tradeoff attack, which typically has time complexity $O(2^{0.5L})$, memory complexity $O(2^{0.5L})$ by using $O(2^{0.25L})$ -bit keystream after a pre-computation phase of complexity $O(2^{0.75L})$.

In [22], a simple method of reducing the key space is introduced and the entropy leakage analysis shows that the average key space of the self-shrinking generator is $O(2^{0.75L})$. A faster cryptanalysis of the SSG is proposed by Mihaljević in [26] with time complexity varying from $O(2^{0.5L})$ to $O(2^{0.75L})$ and the required keystream length ranging from $2^{0.5L}$ to $2^{0.25L}$ accordingly. To get the best complexity estimation $O(2^{0.5L})$, the intercepted keystream length must be greater than $L/2 \cdot 2^{L/2}$, which is beyond the realistic scope for large value of L . In [31], a search tree algorithm is presented to restore an equivalent state of the LFSR from a short segment of the keystream with time complexity $O(2^{0.694L})$. However, the main bottleneck of the attacks in [31,18] is their unrealistically large requirement of memory. Since the self-shrinking generator uses only one LFSR, the method of reducing the memory complexity in [17] is inapplicable. In 2003, P. Ekdahl et al. showed that certain weak feedback polynomials allow very efficient distinguishing attacks on the SSG [10]. Except for these concrete attacks, there is a general time/memory/data tradeoff attack [3] applicable to all stream ciphers in theory. This kind of attack should be taken into consideration especially when a technique called BSW sampling [3] is applicable to the cipher system. It is known that the sampling resistance of the self-shrinking generator is $2^{-L/4}$, thus the reduced search space is $O(2^{0.75L})$. However, such an attack always has a time-consuming preprocessing phase and requires large amount of memory, which are usually impossible for individual cryptanalysts.

In this paper, we propose a new type of guess-and-determine attack on the self-shrinking generator. The large flexibility inherent in the new attack enables us to handle different attack conditions and requirements smoothly. It has no restriction on the form of the LFSR and can reliably recover the initial state of the LFSR with time complexity $O(2^{0.556L})$, memory complexity $O(L^2)$ from $O(2^{0.161L})$ -bit keystream for $L \geq 100$ and time complexity $O(2^{0.571L})$, memory complexity $O(L^2)$ from $O(2^{0.194L})$ -bit keystream for $L < 100$. Compared with the general time/memory/data tradeoff attack, our attack avoids the time-consuming pre-computation phase and the large memory requirement in the TMD attack, while without a substantial compromise of the real processing complexity. Comparisons with other known attacks against the self-shrinking

generator show that our attack offers the best tradeoff between the complexities (time, memory and pre-computation) and the required keystream length. Therefore, our result is the best answer to the open problem in STORK project known so far.

The rest of this paper is organized as follows. We present a detailed description of our attack in Section 2 with theoretical analysis. In Section 3, experimental results to verify the feasibility of our attack and comprehensive comparisons with the previously known attacks on the self-shrinking generator are provided. Finally, some conclusions are given in Section 4.

2 Our Attack

The aim of our attack is to restore the initial state or an equivalent initial state of the LFSR used in the self-shrinking generator from a keystream segment of realistic length. We first state some basic facts on the self-shrinking generator and on the underlying maximal length sequences, then the guess-and-determine attack is presented in detail followed by the theoretical complexity analysis.

2.1 Basic Facts

Let $a = a_0, a_1, \dots$ be the maximal length sequence produced by LFSR A used in the self-shrinking generator and $z = z_0, z_1, \dots$ be the keystream. First note that the two decimated sequences $a_0, a_2, \dots, a_{2i}, \dots$ and $a_1, a_3, \dots, a_{2i+1}, \dots$ are shift equivalent to the original sequence a [13]. They share the same feedback polynomial as that of sequence a and differ only by some shift. The following lemma determines the shift value between sequence $\{a_{2i}\}$ and $\{a_{2i+1}\}$.

Lemma 1. *Let $a = a_0, a_1, \dots$ be a binary maximal length sequence produced by a LFSR of length L , then the shift value τ between the two decimated sequences $c = \{a_{2i}\}$ and $b = \{a_{2i+1}\}$ is 2^{L-1} , i.e. for each integer $i \geq 0$, $b_i = c_{i+2^{L-1}}$.*

Proof. It suffices to note that $c_{i+2^{L-1}} = a_{2 \cdot (i+2^{L-1})} = a_{2i+2^L} = a_{2i+1+2^L-1} = a_{2i+1} = b_i$.

Lemma 1 shows the exact shift value between $\{a_{2i}\}$ and $\{a_{2i+1}\}$, which will facilitate the determination of the relationship between them. Keep the notations as above, we have the following lemma.

Lemma 2. *Let $f(x) = 1 + c_1x + c_2x^2 + \dots + c_{L-1}x^{L-1} + x^L$ be the primitive feedback polynomial of LFSR A over $GF(2)$, i.e. for each $i \geq 0$, $a_{i+L} = \sum_{j=1}^L c_j a_{i+L-j}$, where $c_L = 1$, then there exists a polynomial $h(x) = \sum_{i=0}^{L-1} h_i x^i$ such that $h(x) \equiv x^\tau \pmod{f^*(x)}$, where $f^*(x)$ is the reciprocal polynomial of $f(x)$ and $\tau = 2^{L-1}$ is the shift value between $c = \{a_{2i}\}$ and $b = \{a_{2i+1}\}$. Besides, the polynomial $h(x)$ can be efficiently computed as illustrated below for very large value of L .*

Proof. The former part of this lemma is a straightforward conclusion according to the theory of maximal sequences [13]. It reveals that

$$b_i = a_{2i+1} = \sum_{j=0}^{L-1} h_j c_{i+j} = \sum_{j=0}^{L-1} h_j a_{2(i+j)}, \quad (1)$$

i.e. each b_i is a linear combination of some c_i .

We follow the following recursive procedures to compute $h(x)$. More precisely, the linear coefficients h_j can be determined by recursively computing $x^i \bmod f^*(x) = x(x^{i-1} \bmod f^*(x)) \bmod f^*(x)$ for moderately large L . For very large value of L , this can be fulfilled by the combination of the recursive procedure with the following small step strategy, i.e. we first determine a set of values $\{\tau_1, \dots, \tau_t\}$ such that

$$x^{2^{L-1}} \bmod f^*(x) = x^{\prod_{j=1}^t \tau_j} \bmod f^*(x) = ((x^{\tau_1} \bmod f^*(x))^{\tau_2} \dots)^{\tau_t} \bmod f^*(x),$$

where $\prod_{j=1}^t \tau_j = 2^{L-1}$ and each τ_j is chosen so that $x^{\tau_j} \bmod f^*(x)$ can be computed efficiently by the available method such as the Square-and-Multiply method [20] in rational time. Hence, the linear coefficients h_j can be computed in an acceptable time for very large L in this way.

Table 1 lists the corresponding $h(x)$, obtained by the above combination method, of some primitive polynomials of length up to 300. Here we use $\tau_i = 2^{10}$ for $i = 1, \dots, \lceil (L-1)/10 \rceil - 1$ and $\tau_{\lceil (L-1)/10 \rceil} = 2^{L-1-10 \cdot (\lceil (L-1)/10 \rceil - 1)}$ so that even $x^{2^{299}} \bmod f^*(x)$ with $f(x)$ being a primitive polynomial of degree 300 can be computed in about one hour on a Pentium 4 Processor. This completes the proof.

Lemma 2 shows that compared with the real attack complexity $O(2^{0.556L})$ or $O(2^{0.571L})$, the complexity of computing the linear relationship between $\{a_{2i}\}$ and $\{a_{2i+1}\}$ is negligible. The overall complexity of our attack is dominated by the complexity of the guess-and-determine algorithm given below.

2.2 The Guess-and-Determine Algorithm

The basic idea of a guess-and-determine attack on a stream cipher is to guess some bits of the internal state and derive other bits of the internal state through the relationship between the keystream bits and the internal state bits introduced by the keystream generation process. The validity of a guessed and determined internal state is checked by running the cipher forward from that state. If the generated keystream matches the intercepted keystream, we accept it. Otherwise, we discard the current candidate and try the attack again to get new state candidates.

Oppositely to the methods in other articles, here we do not directly apply the guess-and-determine idea to sequence $\{a_i\}$. Instead we consider the decimated sequence $\{a_{2i}\}$. With the knowledge of $\{a_{2i}\}$, $\{a_i\}$ can be easily recovered from simple linear algebra.

Table 1. Computational results of $h(x)$ on a Pentium 4 processor using Mathematica with the above combination method

$f(x)$	$x^{2^{L-1}} \bmod f^*(x)$
$1 + x + x^{37} + x^{38} + x^{80}$	$x^2 + x^4 + x^5 + x^6 + x^7 + x^{11}$ $+x^{14} + x^{15} + x^{17} + x^{19} + x^{20}$ $+x^{21} + x^{23} + x^{24} + x^{25} + x^{29}$ $+x^{32} + x^{33} + x^{35} + x^{37} + x^{38}$ $+x^{39} + x^{41} + x^{44} + x^{45} + x^{46}$ $+x^{47} + x^{51} + x^{54} + x^{55} + x^{57}$ $+x^{59} + x^{60} + x^{62} + x^{63} + x^{64}$ $+x^{65} + x^{69} + x^{72} + x^{73} + x^{75}$ $+x^{77} + x^{78} + x^{79}$
$1 + x^{37} + x^{100}$	$x^{19} + x^{32} + x^{69}$
$1 + x^2 + x^{15} + x^{17} + x^{168}$	$x^8 + x^{76} + x^{77} + x^{91} + x^{92}$
$1 + x^7 + x^{18} + x^{36} + x^{83} + x^{130} + x^{206} + x^{253} + x^{300}$	$x^6 + x^9 + x^{11} + x^{16} + x^{21} + x^{23}$ $+x^{24} + x^{25} + x^{26} + x^{30} + x^{32}$ $+x^{33} + x^{34} + x^{35} + x^{36} + x^{37}$ $+x^{38} + x^{41} + x^{43} + x^{44} + x^{45}$ $+x^{46} + x^{54} + x^{55} + x^{56} + x^{57}$ $+x^{60} + x^{65} + x^{68} + x^{70} + x^{71}$ $+x^{75} + x^{76} + x^{78} + x^{80} + x^{82}$ $+x^{83} + x^{84} + x^{85} + x^{87} + x^{89}$ $+x^{91} + x^{92} + x^{93} + x^{94} + x^{95}$ $+x^{96} + x^{97} + x^{98} + x^{102}$ $+x^{104} + x^{105} + x^{107} + x^{109}$ $+x^{110} + x^{112} + x^{113} + x^{115}$ $+x^{118} + x^{120} + x^{122} + x^{125}$ $+x^{126} + x^{128} + x^{129} + x^{136}$ $+x^{139} + x^{141} + x^{146} + x^{147}$ $+x^{151} + x^{153} + x^{154} + x^{155}$ $+x^{156} + x^{160} + x^{162} + x^{163}$ $+x^{164} + x^{165} + x^{166} + x^{167}$ $+x^{168} + x^{171} + x^{173} + x^{174}$ $+x^{175} + x^{179} + x^{181} + x^{183}$ $+x^{184} + x^{185} + x^{186} + x^{187}$ $+x^{188} + x^{190} + x^{191} + x^{196}$ $+x^{200} + x^{201} + x^{203} + x^{204}$ $+x^{209} + x^{213} + x^{214} + x^{215}$ $+x^{216} + x^{217} + x^{218} + x^{219}$ $+x^{220} + x^{228} + x^{231} + x^{232}$ $+x^{233} + x^{238} + x^{239} + x^{241}$ $+x^{243} + x^{245} + x^{246} + x^{247}$ $+x^{248} + x^{252} + x^{254} + x^{255}$ $+x^{256} + x^{257} + x^{258} + x^{260}$ $+x^{263} + x^{265} + x^{266} + x^{267}$ $+x^{270} + x^{273} + x^{276} + x^{277}$ $+x^{282} + x^{289} + x^{290} + x^{291}$ $+x^{295} + x^{296} + x^{298} + x^{299}$

More precisely, to attack a self-shrinking generator, we first guess a l -bit length segment

$$A_0^{l-1} = (a_0, a_2, \dots, a_{2(l-1)}) \tag{2}$$

of the initial state $(a_0, a_2, \dots, a_{2(L-1)})$ of $\{a_{2i}\}$, as shown in Figure 1, thus there are $L - l$ bits (black points in Figure 1) of the initial state left unknown. Let $W_H(\cdot)$ be the hamming weight of the corresponding vector, then from the guessed segment, we can get $W_H(A_0^{l-1})$ linear equations on the remaining $L - l$ bits via the shift structure (illustrated by arrowhead in Figure 1). For example, if $a_{2i} = 1$ ($0 \leq i \leq l - 1$), then we have

$$b_i = a_{2i+1} = \sum_{j=0}^{L-1} h_j a_{2(i+j)} = \sum_{j=0}^{l-1} h_j a_{2(i+j)} + \sum_{j=l}^{L-1} h_j a_{2(i+j)} = z_{\sum_{j=0}^{l-1} a_{2i}}, \tag{3}$$

where $h(x) = \sum_{j=0}^{L-1} h_j x^j$ is the polynomial satisfying $h(x) \equiv x^{2^{L-1}} \pmod{f^*(x)}$ found by Lemma 2. Note that the partial sum $\sum_{j=0}^{l-1} h_j a_{2(i+j)}$ in (3) is a known



Fig. 1. Guess-and-determine process

parameter because we guessed the value of $(a_0, a_2, \dots, a_{2(l-1)})$, thus (3) is a linear equation on $L - l$ variables $(a_{2l}, \dots, a_{2(L-1)})$. Once there is a bit $a_{2i} = 1$ for $0 \leq i \leq l - 1$, we will have one linear equation on $(a_{2l}, \dots, a_{2(L-1)})$. Our observation is that the more 1 in the guessed segment A_0^{l-1} , the more linear equations on the remaining $L - l$ bits we can get. The extreme case is that if $(a_0, a_2, \dots, a_{2(l-1)}) = (1, 1, \dots, 1)$, then we will have l linear equations on $L - l$ variables. In order to get an efficient attack, here we do not exhaustively search over all the possible values of A_0^{l-1} . Instead, we just search over those possible values of A_0^{l-1} satisfying (without loss of generality, we assume $a_0 = 1$)

$$W_H(A_0^{l-1}) \geq \lceil \alpha \cdot l \rceil, \tag{4}$$

where $\lceil x \rceil$ gives the smallest integer greater than or equal to x and α ($0.5 \leq \alpha \leq 1$) is a parameter to be determined later. Hence, we can get at least $\lceil \alpha \cdot l \rceil$ linear equations on the remaining $L - l$ bits by this method.

Now a crucial question arises naturally, i.e. how about the linear dependency of these linear equations? Fortunately, from the initial state $(a_0, a_2, \dots, a_{2(L-1)})$ of $\{a_{2i}\}$, we have

$$(a_0, \dots, a_{2(L-1)}, a_{2L}, \dots, a_{2(N-1)}) = (a_0, a_2, \dots, a_{2(L-1)}) \cdot G,$$

where N is the length of sequence $\{a_{2i}\}$ under consideration and G is a $L \times N$ matrix over $GF(2)$:

$$G = \begin{pmatrix} g_0^0 & g_1^0 & \cdots & g_{N-1}^0 \\ g_0^1 & g_1^1 & \cdots & g_{N-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{L-1} & g_1^{L-1} & \cdots & g_{N-1}^{L-1} \end{pmatrix},$$

i.e. each a_{2i} is a linear combination of $(a_0, a_2, \dots, a_{2(L-1)})$. Since for each $i \geq 0$, $a_{2i+1} = a_{2i+2L-1}$, the column vectors $g_i = (g_i^0, g_i^1, \dots, g_i^{L-1})^T$ corresponding to the bits selected in $(a_1, a_3, \dots, a_{2l-1})$ according to the pattern of $(a_0, a_2, \dots, a_{2(L-1)})$ can be regarded as random vectors over $GF(2)^L$. Thus, this holds also for the truncated versions of g_i over $GF(2)^{L-l}$ which form the coefficient matrix on the remaining $L-l$ unknown bits. The following lemma guarantees that the matrix formed by the truncated random column vectors always has the rank close to its maximum.

Lemma 3. ([30]) *The probability that a random generated $m \times n$ binary matrix has rank r ($1 \leq r \leq \min(m, n)$) is*

$$P_r = 2^{r(m+n-r)-nm} \prod_{i=0}^{r-1} \frac{(1-2^{i-m})(1-2^{i-n})}{1-2^{i-r}}. \quad (5)$$

Although we can sometimes get more than $\lceil \alpha l \rceil$ linear equations by the above searching method, we only use the lower bound $\lceil \alpha l \rceil$ in the estimation of the linear independent equations and let $\lceil \alpha \cdot l \rceil = L-l$. The reason for doing so is to derive the worst case complexity of our guess-and-determine algorithm in the Section 2.3. By lemma 3, the probability that a random generated $\lceil \alpha l \rceil \times (L-l)$ binary matrix has rank $r \geq \lceil \alpha l \rceil - 5$ is

$$P(r \geq \lceil \alpha l \rceil - 5) = \sum_{r=\lceil \alpha l \rceil - 5}^{\lceil \alpha l \rceil} 2^{-(r-\lceil \alpha l \rceil)(r-L+l)} \prod_{i=0}^{r-1} \frac{(1-2^{i-\lceil \alpha l \rceil})(1-2^{i-L+l})}{1-2^{i-r}}. \quad (6)$$

Simulation results show that $P(r \geq \lceil \alpha l \rceil - 5) \geq 0.99$ for $L \leq 1500$, i.e. the linear equations we get are almost linear independent. We can compensate the linear dependency of the linear system by an exhaustive search at a small scale.

The entire description of the guess-and-determine attack (algorithm A) is as follows (in C-like notation).

- **Parameter:** α, L
- **Input:** keystream $\{z_i\}_{i=0}^{N-1}$, feedback polynomial $f(x)$
- **Processing:**
 1. Apply the combination strategy illustrated in Section 2.1 to compute $x^{2^{L-1}} \bmod f^*(x)$, where $f^*(x)$ is the reciprocal polynomial of $f(x)$
 2. **for** all l -bit segment A_0^{l-1} satisfying (4) **do**
 - **for** $k=0$ **to** $l-1$ **do**

```

* if  $a_{2k} = 1$  then
    Using  $h(x)$  obtained in step 1 and  $f(x)$ , derive a linear expression
    on the remaining bits in  $A_t^{L-1} = (a_{2l}, \dots, a_{2(L-1)})$  and store the
    expression in matrix  $U$ 
end if
end for
• for  $j = 0$  to  $N - 1 - \lceil \alpha \cdot l \rceil$  do
    (a) Check the linear consistency [32] of the linear system using keystream
    indexed from  $z_j$ 
    (b) if the linear consistency test is OK then
        * Solve the linear system in  $U$  according to the keystream indexed
        from  $z_j$  to get a state candidate  $(a'_0, a'_2, \dots, a'_{2(L-1)})$  or a small list
        of candidates
        * for each candidate state do
            i. Run the SSG forward from the candidate state and check the
            generated keystreams with  $\{z_i\}_{i=j}^{N-1}$ 
            ii. if the correlation test is OK then
                Output that candidate and break the loop
                else continue
                end if
            end for
        else continue
        end if
    end for
end for
– Output: the initial state or an equivalent state  $(a_0, a_2, \dots, a_{2(L-1)})$ 

```

Here the *for* loop works in the same way as in C language. Assume we start with the keystream $\{z_i\}_{i=0}^{N-1}$. We first derive the linear expressions as in (3) from the guessed segment A_0^{l-1} , then associate them with the keystream indexed from z_0 and test the linear consistency of the resulting system. If the test fails, then try the keystream indexed from z_1 , indexed from z_2, \dots , and so on. If we cannot get a consistent linear system based on the keystream in hand, discard the current guess of A_0^{l-1} and try another guess to restart. If we find it, solve the system to get a candidate state $(a'_0, a'_2, \dots, a'_{2(L-1)})$ or a small list of candidate states. Run the self-shrinking generator forward from each candidate state and generate the corresponding keystream. If the generated keystream does not match the intercepted keystream, discard that candidate and try another one. If all the candidates failed to find a match, then try another guess of A_0^{l-1} to restart the above whole process. If enough keystream is available, we expect to find the initial state (or an equivalent state) corresponding to the intercepted keystream with high success probability.

2.3 Complexity Analysis

Now we analyze the time, memory and data complexity of the algorithm A. We first establish the basic equation of our attack. Then, the corresponding time, memory and data complexity are derived in the most general case, respectively.

Finally, we discuss the success rate of the algorithm A and point out the optimal choices of the attack parameters.

From algorithm A, to cover the $L - l$ unknown bits by $O(\alpha \cdot l)$ linear independent equations, we let

$$O(\alpha \cdot l) = L - l \implies l = O\left(\frac{1}{1 + \alpha} \cdot L\right). \tag{7}$$

Since we just want to derive the magnitude, here we ignore the possible small number of linear dependent equations.

In algorithm A, we only search over those possible values of A_0^{l-1} that satisfy (4). Let $H = \{A_0^{l-1} \mid \lceil \alpha l \rceil \leq W_H(A_0^{l-1}) \leq l \text{ and } a_0 = 1\}$, then

$$|H| = \sum_{i=\lceil \alpha l \rceil - 1}^{l-1} \binom{l-1}{i},$$

where $|\cdot|$ denotes the cardinality of a set. The proportion between the l -bit values contained in $|H|$ and all the 2^l possible values is $\frac{|H|}{2^l}$, we rewrite it as

$$\frac{\sum_{i=\lceil \alpha l \rceil - 1}^{l-1} \binom{l-1}{i}}{2^l} = \frac{2^{\beta l}}{2^l} = 2^{-(1-\beta) \cdot l}, \tag{8}$$

where β is a parameter determined by α and l . From (8), we have

$$\beta = \frac{1}{l} \cdot \log_2 \sum_{i=\lceil \alpha l \rceil - 1}^{l-1} \binom{l-1}{i}. \tag{9}$$

Combining with (7), we have a function $\beta = \beta(\alpha, L)$, as shown in Figure 2. It is worth noting that β decreases with α increasing.

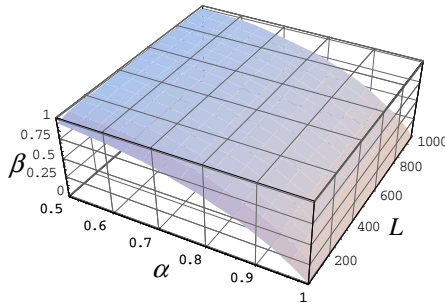


Fig. 2. β as a function of α and the LFSR length L

For the algorithm A to succeed, we must find at least one match pair between the state set H and the keystream segments involved in algorithm A. Assume sequence $\{a_i\}$ is purely random (consisting of independent and uniformly distributed binary random variables), thus the keystream length N should satisfy

$$(N - L) \cdot \sum_{i=\lceil \alpha l \rceil - 1}^{l-1} \binom{l-1}{i} \left(\frac{1}{2}\right)^{l-1} \geq 1,$$

i.e.

$$N > \frac{2^{l-1}}{\sum_{i=\lceil \alpha l \rceil - 1}^{l-1} \binom{l-1}{i}} = \frac{2^{l-1}}{2^{\beta \cdot l}} = 2^{(1-\beta) \cdot l-1} \implies N \sim O(2^{\frac{1-\beta}{1+\alpha} \cdot L}). \quad (10)$$

Algorithm A searches over the state set H and at each iteration, it checks along the keystream $\{z_i\}_{i=0}^{N-1}$ to find the suited segment. Therefore, the worst case time complexity is

$$O(N - L) \cdot O(2^{\beta \cdot l}) = O(2^{\frac{1}{1+\alpha} \cdot L}). \quad (11)$$

The following theorem summarizes the above results.

Theorem 1. *Keep the notations as above. The guess-and-determine algorithm A in section 2.2 has time complexity $O(L^3 \cdot 2^{\frac{1}{1+\alpha} \cdot L})$, memory complexity $O(L^2)$ and data complexity $O(2^{\frac{1-\beta}{1+\alpha} \cdot L})$, where L is the length of the LFSR used in the SSG, $0.5 \leq \alpha \leq 1$, β is a parameter determined by α and L .*

Proof. For the time complexity, note (11) and that in each iteration of algorithm A, we have to check the linear consistency of the linear system and then solve it. This contributes the L^3 factor to time complexity. For the memory complexity, it suffices to note that in the algorithm A, we only need to store the matrix U corresponding to the current guess of A_0^{l-1} and the memory usage in step 2 is dominating. The data complexity follows (10).

Corollary 1. *Keep the notations as those in Theorem 1 and under the above complexities, the success probability of algorithm A is*

$$P_{succ} = 1 - (1 - 2 \cdot 2^{-\frac{1-\beta}{1+\alpha} \cdot L})^{N-L},$$

where N is the length of the keystream used in the attack.

Proof. It suffices to note that in algorithm A, we totally check $N - L$ keystream segments and each segment matches to a state in H with probability $2 \cdot 2^{-\frac{1-\beta}{1+\alpha} \cdot L}$.

To get the optimal performance of our attack, we should optimize the parameters α and β of the algorithm A. Table 2 lists the asymptotic time, memory and data complexities corresponding to the different choices of α with the LFSR length $L \geq 100$. It is worth noting that the values of β are just approximations. In a real attack, we recommend using (7) and (9) to compute the more accurate values. (In Table 2, 3 and 4, we ignore the polynomial factors in the corresponding time complexities of these attacks, e.g. for the attack in [31], this factor is L^4 and for the BDD-based attack in [18], this factor is $L^{O(1)}$). To beat the general time/memory/data tradeoff attack, we recommend using $\alpha = 0.8$. Accordingly, the asymptotic time, memory and data complexities are $O(2^{0.556L})$, $O(L^2)$ and $O(2^{0.161L})$, respectively.

Table 2. The asymptotic time, memory and data complexities of algorithm A corresponding to different choices of α ($L \geq 100$)

α	β	Time	Memory	Data
0.5	0.99	$O(2^{0.667L})$	$O(L^2)$	$O(2^{0.007L})$
0.6	0.96	$O(2^{0.625L})$	$O(L^2)$	$O(2^{0.025L})$
0.75	0.80	$O(2^{0.571L})$	$O(L^2)$	$O(2^{0.114L})$
0.8	0.71	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.161L})$
0.9	0.46	$O(2^{0.526L})$	$O(L^2)$	$O(2^{0.284L})$
1.0	0.00	$O(2^{0.5L})$	$O(L^2)$	$O(2^{0.5L})$

Table 3. The time, memory and data complexities of algorithm A corresponding to different choices of α ($40 \leq L < 100$)

α	β	Time	Memory	Data
0.5	0.93	$O(2^{0.667L})$	$O(L^2)$	$O(2^{0.047L})$
0.6	0.88	$O(2^{0.625L})$	$O(L^2)$	$O(2^{0.075L})$
0.75	0.66	$O(2^{0.571L})$	$O(L^2)$	$O(2^{0.194L})$
0.8	0.57	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.239L})$
0.9	0.36	$O(2^{0.526L})$	$O(L^2)$	$O(2^{0.337L})$
1.0	0.00	$O(2^{0.5L})$	$O(L^2)$	$O(2^{0.5L})$

Note that the values listed in Table 2 are asymptotic. For $40 \leq L < 100$, the corresponding values are listed in Table 3. To beat the TMD attack with $40 \leq L < 100$, we recommend using $\alpha = 0.75$ or $\alpha = 0.8$. In both cases, the corresponding memory and data complexities are better than those of the TMD attack, while without a substantial compromise of the time complexity.

3 Comparisons and Experimental Results

We first present a detailed comparison with some other well-known attacks against the self-shrinking generator. Then, a number of experimental results are provided to verify the actual performance of the new attack. The advantages of our attack are pointed out at the end of this section.

3.1 Comparisons with Other Attacks

We mainly focus on the following attacks against the self-shrinking generator, i.e. the Mihaljević's attack in [26], the search tree attack in [31], the BDD-based attack in [18] and the time/memory/data tradeoff attack in [3]. Table 4 summarizes the corresponding results.

We can see from Table 4 that our attack achieves the best tradeoff between the time, memory, data and pre-computation complexities. More precisely, The attack in [26] suffers from the large amount of the keystream, which reaches $O(2^{0.5L})$ to obtain the best time complexity $O(2^{0.5L})$. Both the search tree attack

Table 4. Asymptotic complexity comparisons with some other well-known attacks against the self-shrinking generator with the LFSR of length L

Attack	Pre-computation	Time	Memory	Data
[26]A	-	$O(2^{0.5L})$	$O(L)$	$O(2^{0.5L})$
[26]B	-	$O(2^{0.75L})$	$O(L)$	$O(2^{0.25L})$
[31]	-	$O(2^{0.694L})$	$O(2^{0.694L})$	$O(L)$
[18]	-	$O(2^{0.656L})$	$O(2^{0.656L})$	$O(L)$
[3]A	$O(2^{0.75L})$	$O(2^{0.5L})$	$O(2^{0.5L})$	$O(2^{0.25L})$
[3]B	$O(2^{0.67L})$	$O(2^{0.67L})$	$O(2^{0.33L})$	$O(2^{0.33L})$
Ours ($\alpha = 0.5$)	-	$O(2^{0.667L})$	$O(L^2)$	$O(2^{0.007L})$
Ours ($\alpha = 0.75$)	-	$O(2^{0.571L})$	$O(L^2)$	$O(2^{0.114L})$
Ours ($\alpha = 0.8$)	-	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.161L})$

in [31] and the BDD-based attack in [18] are unrealistic in terms of the memory requirement. In addition, the data complexity of our attack with $\alpha = 0.5$ are in the same order as those in [31] and [18] for the LFSR length L up to 2000. The two typical TMD attacks are derived according to the two points $T = N^{2/3}$, $M = D = N^{1/3}$ and $T = M = N^{1/2}$, $D = N^{1/4}$ on the curve $TM^2D^2 = N^2$ with pre-computation $P = N/D$, where T , M , D , N denote time, memory, data and search key space, respectively. Even regardless of the heavy pre-computation phase of the TMD attack, our attack with $\alpha = 0.8$ has much better memory and data complexity compared with the two TMD attacks, while without a substantial compromise of the real time complexity.

On the other hand, our attack can deal with different attack conditions and requirements smoothly due to the flexible choices of α . If only very short keystream and very limited disk space are available to the attacker, we still can launch a guess-and-determine attack successfully against the SSG with $\alpha \leq 0.6$. In this way, we avoid the large memory requirement of the two attacks in [31] and [18].

3.2 Experimental Results

We made a number of experimental results in C language on a Pentium 4 processor to check the actual performance of our attack.

Since the guess-and-determine attack in Section 2.2 has no restriction on the LFSR form, it has been implemented and tested many times for random chosen initial states and primitive polynomials of degree $10 \leq L \leq 50$ involved in the self-shrinking generator. For $10 \leq L \leq 40$, we use $\alpha = 0.6$ to mount the attack on the self-shrinking generator. For $40 < L \leq 50$, we use $\alpha = 0.8$. The results are rather satisfactory. The required keystream length are very close to the theoretical value in magnitude and the time complexity seems to be upper bounded by the theoretical value, which is just in expectation.

For example, let the LFSR's feedback polynomial be $f(x) = 1 + x^2 + x^{19} + x^{21} + x^{40}$, then the shift value is $x^{239} \bmod f^*(x) = x^{11} + x^{29} + x^{30}$, where $f^*(x)$ is the reciprocal polynomial of $f(x)$. For a random chosen initial state, our attack

takes several minutes to recover the initial state or an equivalent state with success rate (see Table 3 and Corollary 1)

$$1 - (1 - 2 \cdot 2^{-(1-0.88) \cdot 40 / (1+0.6)})^{(200-40)} > 0.99$$

from 200 bits keystream.

As a summary, our attack has at least the following advantages over the past relevant attacks against the self-shrinking generator:

- significantly smaller memory complexity with the time complexity quite close to $O(2^{0.5L})$.
- no pre-computation or if like (pre-compute $h(x)$), significantly smaller pre-processing time complexity without a compromise of the real attack complexity.
- flexibility to different attack conditions and requirements

These features guarantee that the proposed guess-and-determine attack can provide a better tradeoff between the time, memory and data complexities than all the previously known attacks against the self-shrinking generator. Especially, it compares favorably with the general time/memory/data tradeoff attack. Thus, our attack is the best answer known so far to a well-known open problem specified by the European STORK project.

4 Conclusions

In this paper, we proposed a new type of guess-and-determine attack on the self-shrinking generator. The new attack adapts well to different attack conditions and enables us to analyze the self-shrinking generator with the best tradeoff between the time, memory, data and pre-computation complexities known so far. So our result is the best answer to the corresponding open problem in STORK project known so far.

Acknowledgements. We would like to thank one of the anonymous reviewers for very helpful comments.

References

1. F. Armknecht, M. Krause, “Algebraic Attacks on Combiner with Memory”, *Advances in Cryptology-Crypto’2003*, LNCS vol. 2729, Springer-Verlag, (2003), pp. 162-175.
2. F. Armknecht, “Improving Fast Algebraic Attacks”, *Fast Software Encryption-FSE’2004*, LNCS, pp. 47-63, February 2004.
3. A. Biryukov, A. Shamir, “Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers”, *Advances in Cryptology-ASIACRYPT’2000*, LNCS vol. 1976, Springer-Verlag, (2000), pp. 1-13.

4. A. Canteaut, M. Trabbia, "Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5", *Advances in Cryptology-EUROCRYPT'2000*, LNCS vol. 1807, Springer-Verlag, (2000), pp. 573-588.
5. P. Chose, A. Joux, M. Mitton, "Fast Correlation Attacks: An Algorithmic Point of View", *Advances in Cryptology-EUROCRYPT'2002*, LNCS vol. 2332, Springer-Verlag, (2002), pp. 209-221.
6. S.R. Blackburn, "The linear complexity of the self-shrinking generator", *IEEE Transactions on Information Theory*, 45(6), September 1999, pp. 2073-2077.
7. D. Coppersmith, H. Krawczyk, Y. Mansour, "The Shrinking Generator", *Advances in Cryptology-Crypto'93*, LNCS vol. 773, Springer-Verlag, (1994), pp.22-39.
8. N. T. Courtois, "Fast Algebraic Attacks on Stream ciphers with Linear Feedback", *Advances in Cryptology-Crypto'2003*, LNCS vol. 2729, Springer-Verlag, (2003), pp. 176-194.
9. N. T. Courtois, W. Meier, "Algebraic Attacks on Stream ciphers with Linear Feedback", *Advances in Cryptology-EUROCRYPT'2003*, LNCS vol. 2656, Springer-Verlag, (2003), pp. 345-359.
10. P. Ekdahl, T. Johansson and W. Meier, "A note on the self-shrinking generator", *Proceeding of IEEE symposium on Information Theory*, (2003), pp. 166.
11. J. Dj. Golić, "Embedding and probabilistic correlation attacks on clock-controlled shift registers", *Advances in Cryptology-EUROCRYPT'94*, LNCS vol. 950, Springer-Verlag, (1994), pp. 230-243.
12. J. Dj. Golić, "Correlation analysis of the shrinking Generator", *Advances in Cryptology-Crypto'2001*, LNCS vol. 2139 Springer-Verlag, (2001), pp. 440-457.
13. S. W. Golomb, "*Shift Register Sequences*", Aegean Park Press, Laguna Hills(CA), Revised edition, 1982.
14. T. Johansson, "Reduced complexity correlation attacks on two clock-controlled generators", *Advances in Cryptology-ASIACRYPT'98*, LNCS vol. 1514, Springer-Verlag, (1998), pp. 342-357.
15. T. Johansson, F. Jonsson, "Improved fast correlation attack on stream ciphers via convolutional codes", *Advances in Cryptology-EUROCRYPT'1999*, LNCS vol. 1592, Springer-Verlag, (1999), pp. 347-362.
16. T. Johansson, F. Jönsson, "Fast correlation attacks through reconstruction of linear polynomials", *Advances in Cryptology-Crypto'2000*, LNCS vol. 1880, Springer-Verlag, (2000), pp. 300-315.
17. M. Krause and D. Stegemann, "Reducing the Space Complexity of BDD-based Attacks on Keystream Generators", *Fast Software Encryption-FSE'2006*, to appear.
18. M. Krause, "BDD-based cryptanalysis of keystream generators", *Advances in Cryptology-EUROCRYPT'2002*, LNCS vol. 2332, Springer-Verlag, (2002), pp. 222-237.
19. H. Krawczyk, "The shrinking generator: Some practical considerations", *Fast Software Encryption-FSE'94*, LNCS vol. 809, Springer-Verlag, (1994), pp. 45-46.
20. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
21. J. L. Massey, "Shift register synthesis and BCH decoding", *IEEE Transactions on Information Theory*, 15. 1969, pp. 122-127.
22. W. Meier, and O. Staffelbach, "The Self-Shrinking generator", *Advances in Cryptology-EUROCRYPT'1994*, LNCS vol. 950, Springer-Verlag, (1995), pp. 205-214.
23. W. Meier, O. Staffelbach, "Fast correlation attacks on certain stream ciphers", *Journal of Cryptology*, (1989) 1, pp. 159-176.

24. M. Mihaljević, P.C. Fossorier, H.Imai, “Fast correlation attack algorithm with list decoding and an application”, *Fast Software Encryption-FSE’2001*, LNCS vol. 2355, Springer-Verlag, (2002), pp. 196-210.
25. M. Mihaljević, P.C. Fossorier, H.Imai, “A Low-complexity and high-performance algorithm for fast correlation attack”, *Fast Software Encryption-FSE’2000*, LNCS vol. 1978, Springer-Verlag, (2001), pp. 196-212.
26. M. J. Mihaljević, “A faster cryptanalysis of the self-shrinking generator”, *ACISP’96*, LNCS vol. 1172, Springer-Verlag, (1998), pp. 147-158.
27. L. Simpson, J. Dj. Golić, “A probabilistic correlation attack on the shrinking generator”, *ACISP’98*, LNCS vol. 1438, Springer-Verlag, (1996), pp. 182-189.
28. I. Shparlinski, “On some properties of the shrinking generator”, <http://www.comp.mq.edu.au/~igor/Shrink.ps>.
29. STORK project, <http://www.stork.eu.org/documents/RUB-D6-2-1.pdf>
30. Z. Wan, “*Geometry of Classical Groups over Finite Fields*”, Science Press, New York, Second edition, 2002.
31. E. Zenner, M. Krause, S. Lucks, “Improved Cryptanalysis of the Self-Shrinking Generator”, *Proc. ACISP’2001*, LNCS vol. 2119, Springer-Verlag, (2001), pp. 21-35.
32. K. Zeng, C. Yang, T. Rao, “On the linear consistency test (LCT) in cryptanalysis with applications”, *Advances in Cryptology-Crypto’1989*, LNCS vol. 435, Springer-Verlag, (1989), pp. 164-174.

On the (In)security of Stream Ciphers Based on Arrays and Modular Addition*

Souradyuti Paul and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/COSIC,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium
{Souradyuti.Paul, Bart.Preneel}@esat.kuleuven.be

Abstract. Stream ciphers play an important role in symmetric cryptology because of their suitability in high speed applications where block ciphers fall short. A large number of fast stream ciphers or pseudorandom bit generators (PRBG's) can be found in the literature that are based on arrays and simple operations such as modular additions, rotations and memory accesses (e.g. RC4, RC4A, Py, Py6, ISAAC etc.). This paper investigates the security of array-based stream ciphers (or PRBG's) against certain types of distinguishing attacks in a unified way. We argue, counter-intuitively, that the most useful characteristic of an array, namely, the association of array-elements with unique indices, may turn out to be the origins of distinguishing attacks if adequate caution is not maintained. In short, an adversary may attack a cipher simply exploiting the dependence of array-elements on the corresponding indices. Most importantly, the weaknesses are not eliminated even if the indices and the array-elements are made to follow uniform distributions separately. Exploiting these weaknesses we build distinguishing attacks with reasonable advantage on five recent stream ciphers (or PRBG's), namely, Py6 (2005, Biham *et al.*), IA, ISAAC (1996, Jenkins Jr.), NGG, GGHN (2005, Gong *et al.*) with data complexities $2^{68.61}$, $2^{32.89}$, $2^{16.89}$, $2^{32.89}$ and $2^{32.89}$ respectively. In all the cases we worked under the assumption that the key-setup algorithms of the ciphers produced uniformly distributed internal states. We only investigated the mixing of bits in the keystream generation algorithms. In hindsight, we also observe that the previous attacks on the other array-based stream ciphers (e.g. Py, etc.), can also be explained in the general framework developed in this paper. We hope that our analyses will be useful in the evaluation of the security of stream ciphers based on arrays and modular addition.

1 Introduction

Stream ciphers are of paramount importance in fast cryptographic applications such as encryption of streaming data where information is generated at a high

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

speed. Unfortunately, the state-of-the art of this type of ciphers, to euphemize, is not very promising as reflected in the failure of the NESSIE project to select a single cipher for its profile [13] and also the attacks on a number of submissions for the ongoing ECRYPT project [6]. Because of plenty of common features as well as dissimilarities, it is almost impossible to classify the entire gamut of stream ciphers into small, well-defined, disjoint groups, so that one group of ciphers can be analyzed in isolation of the others. However, in view of the identical data structures and similar operations in a number of stream ciphers and the fact that they are vulnerable against certain kinds of attacks originating from some basic flaws inherent in the design, it makes sense to scrutinize the class of ciphers in a unified way. As the title suggests, the paper takes a closer look at stream ciphers connected by a common feature that each of them uses (i) one or more arrays¹ as the *main* part of the internal state and (ii) the operation *modular addition* in the *pseudo-random bit generation algorithm*. Apart from *addition* over different *groups* (e.g. $\text{GF}(2^n)$ and $\text{GF}(2)$), the stream ciphers under consideration only admit of simple operations such as memory access (direct and indirect) and cyclic rotation of bits, which are typical of any fast stream cipher. In the present discussion we omit the relatively rare class of stream ciphers which may nominally use *array* and *addition*, but their security depends significantly on special functions such as those based on algebraic hard problems, Rijndael S-box etc.

To the best of our knowledge, the RC4 stream cipher, designed by Ron Rivest in 1987, is the first stream cipher which exploits the features of an array in generating pseudorandom bits, using a few simple operations. Since then a large number of array-based ciphers or PRBG's – namely, RC4A [14], VMPC stream cipher [20], IA, IBAA, ISAAC [10], Py [2], Py6 [4], Pypy [3], HC-256 [18], NGG [12], GGHN [8] – have been proposed that are inspired by the RC4 design principles. The Scream family of ciphers [9] also uses arrays and modular additions in their round functions, however, the security of them hinges on a tailor-made function derived from Rijndael S-box rather than mixing of *additions* over different *groups* (e.g., $\text{GF}(2^n)$ and $\text{GF}(2)$) and cyclic rotation of bits; therefore, this family of ciphers is excluded from the class of ciphers to be discussed in the paper.

First, in Table 1, we briefly review the pros and cons of the RC4 stream cipher which is the predecessor of all the ciphers to be analyzed later. Unfortunately, the RC4 cipher is compatible with the old fashioned 8-bit processors only. Except RC4A and the VMPC cipher (which are designed to work on 8-bit processors), all the other ciphers described before are suitable for modern 16/32-bit architectures. Moreover, those 16/32-bit ciphers have been designed with an ambition of incorporating all the positive aspects of RC4, while ruling out its negative properties as listed in Table 1. However, the paper observes that a certain amount of caution is necessary to adapt RC4-like ciphers to 16/32-bit architecture. Here, we mount distinguishing attacks on the ciphers (or PRBG's) Py6, IA, ISAAC, NGG, GGHN – all of them are designed to suit 16/32-bit processors – with data $2^{68.61}$, $2^{32.89}$, $2^{16.89}$, $2^{32.89}$ and $2^{32.89}$ respectively, exploiting

¹ An array is a data structure containing a set of elements associated with unique indices.

Table 1. Pros and cons of the RC4 cipher

Advantages of RC4	Disadvantages of RC4
Arrays allow for huge <i>secret</i> internal state	Not suitable for 16/32-bit architecture
Fast because of fewer operations per round	Several distinguishing attacks
Simple Design	Weak Key-setup algorithm
No key recovery attacks better than brute force	

similar weaknesses in their designs (note that another 32-bit array-based cipher Py has already been attacked in a similar fashion [5,15]). Summarily the attacks on the class of ciphers described in this paper originate from the following basic although not independent facts. However, note that our attacks are based on the assumptions that the key-setup algorithms of the ciphers are ‘perfect’, that is, after the execution of the algorithms they produce uniformly distributed internal states (more on that in Sect. 1.2).

- Array-elements are large (usually of size 16/32 bits), but the array-indices are short (generally of size 8 bits).
- Only a few elements of the arrays undergo changes in consecutive rounds.
- Usage of both pseudorandom index-pointers and pseudorandom array-elements in a round, which apparently seems to provide stronger security than the ciphers with fixed pointers, may leave room for attacks arising from the *correlation* between the index-pointers and the corresponding array-elements (see discussion in Sect. 2.2).
- Usage of simple operations like *addition* over $\text{GF}(2^n)$ and $\text{GF}(2)$ in output generation.

Essentially our attacks based on the above facts have it origins in the *fortuitous states* attack on RC4 by Fluhrer and McGrew [7].

A general framework to attack array-based stream ciphers with the above characteristics is discussed in Sect. 2. Subsequently in Sect. 3.1, 3.2 and 3.3, as concrete proofs of our argument, we show distinguishing attacks on five stream ciphers (or PRBG’s). The purpose of the paper is, by no means, to claim that the array-based ciphers are intrinsically insecure, and therefore, should be rejected without analyzing its merits; rather, we stress that when such a PRBG turns out to be extremely fast – such as Py, Py6, IA, ISAAC, NGG, GGHN – an alert message should better be issued for the designers to recheck that they are free from the weaknesses described here. In Sect. 3.5, we comment on the security of three other array-based ciphers (or PRBG’s) IBAA, Pypy and HC-256 which, for the moment, do not come under attacks, however they are slower than the ones attacked in this paper.

1.1 Notation and Convention

- The symbols \oplus , $+$, $-$, \lll , \ggg , \gg , \ll are used as per convention.
- The i th bit of the variable X is denoted $X_{(i)}$ (the *lsb* is the 0th bit).

- The segment of $m - n + 1$ bits between the m th and the n th bits of the variable X is denoted by $X_{(m,n)}$.
- The abbreviation for Pseudorandom Bit Generator is PRBG.
- $P[A]$ denotes the probability of occurrence of the event A .
- E^c denotes the compliment of the event E .
- At any round t , some part of the internal state is updated before the output generation and the rest is updated after that. Example: in Algorithm 3, the variables a and m are updated before the output generation in line 5. The variables i and b are updated after or at the same time with the output generation. Our convention is: a variable S is denoted by S_t at the time of output generation of round t . As each of the variables is modified in a single line of the corresponding algorithm, after the modification its subscript is incremented.

1.2 Assumption

In this paper we concentrate solely on the *mixing of bits* by the keystream generation algorithms (i.e., PRBG) of several array-based stream ciphers and assume that the corresponding key-setup algorithms are *perfect*. A *perfect* key-setup algorithm produces internal state that leaks no statistical information to the attacker. In other words, because of the *difficulty* of deducing any relations between the inputs and outputs of the key-setup algorithm, the internal state produced by the key-setup algorithm is assumed to follow the uniform distribution.

2 Stream Ciphers Based on Arrays and Modular Addition

2.1 Basic Working Principles

The basic working principle of the PRBG of a stream cipher, based on one or multiple arrays, is shown in Fig. 1. For simplicity, we take snapshots of the internal state, composed of *only* two arrays, at two close rounds denoted by round t and round $t' = t + \delta$. However, our analysis is still valid with more arrays and rounds than just two. Now we delineate the rudiments of the PRBG of such ciphers.

- **Components:** The internal state of the cipher comprises all or part of the following components.
 1. One or more arrays of n -bit elements (X_1 and X_2 in Fig. 1).
 2. One or more variables for indexing into the arrays, i.e., the index-pointers (down arrows in Fig. 1).
 3. One or more random variables usually of n -bit length (m_1, m_2, m'_1, m'_2 in Fig. 1).
- **Modification to the Internal State at a round**
 1. *Index Pointers:* The most notable feature of such ciphers is that it has two sets of index pointers. (i) Some of them are fixed or updated in a *known way*, i.e., independent of the secret part of the state (solid arrows

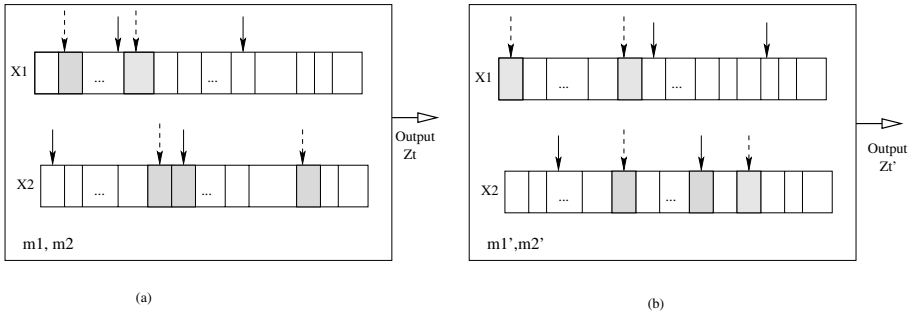


Fig. 1. Internal State at (a) round t and (b) round $t' = t + \delta$

in Fig. 1) and (ii) the other set of pointers are updated *pseudorandomly*, i.e., based on one or more secret components of the internal state (dotted arrows in Fig. 1).

2. *Arrays:* A few elements of the arrays are updated pseudorandomly based on one or more components of the internal state (the shaded cells of the arrays in Fig. 1). Note that, in two successive rounds, only a small number of array-elements (e.g. one or two in each array) are updated. Therefore, most of the array-elements remain identical in consecutive rounds.

3. *Other variables if any:* They are updated using several components of the internal state.

– **Output generation:** The output generation function at a round is a non-linear combination of different components described above.

2.2 Weaknesses and General Attack Scenario

Before assessing the security of array-based ciphers in general, for easy understanding, we first deal with a simple toy-cipher with certain properties which induce distinguishing attack on it. Output at round t is denoted by Z_t .

Remark 1. The basis for the attacks described throughout the paper including the one in the following example is searching for internal states for which the outputs can be predicted with bias. This strategy is inspired by the *fortuitous states* attacks by Fluhrer and McGrew on the RC4 stream cipher [7].

Example 1. Let the size of the *internal state* of a stream cipher with the following properties be k bits.

Property 1. The outputs Z_{t_1}, Z_{t_2} are as follows.

$$Z_{t_1} = X \oplus Y + (A \lll B), \quad (1)$$

$$Z_{t_2} = M + N \oplus (C \lll D) \quad (2)$$

where X, Y, A, B, M, N, C, D are uniformly distributed and independent.

Property 2. [Bias-inducing State] If certain k' bits ($0 < k' \leq k$) of the *internal state* are set to all 0's (denote the occurrence of such state by event E) at round t_1 , then the following equations hold good.

$$X = M, Y = N, B = D = 0, A = C.$$

Therefore, (1) and (2) become

$$Z_{t_1} = X \oplus Y + A, \quad Z_{t_2} = X + Y \oplus A.$$

Now, it follows directly from the above equations that, for a fraction of $2^{-k'}$ of all *internal states*,

$$P[Z_{(0)} = (Z_{t_1} \oplus Z_{t_2})_{(0)} = 0|E] = 1. \quad (3)$$

Property 3. If the *internal state* is chosen randomly from the rest of the states, then

$$P[Z_{(0)} = 0|E^c] = \frac{1}{2}. \quad (4)$$

Combining (3) and (4) we get the overall bias for $Z_{(0)}$,

$$\begin{aligned} P[Z_{(0)} = 0] &= \frac{1}{2^{k'}} \cdot 1 + \left(1 - \frac{1}{2^{k'}}\right) \cdot \frac{1}{2} \\ &= \frac{1}{2} \left(1 + \frac{1}{2^{k'}}\right) \end{aligned} \quad (5)$$

Note that, if the cipher were a secure PRBG then $P[Z_{(0)} = 0] = \frac{1}{2}$. \square

Discussion. Now we argue that an array-based cipher has all the three properties of the above example; therefore, the style of attack presented in the example can possibly be applied to an array-based cipher too. First, we discuss the operations involved in the output generation of the PRBG. Let the internal state consist of N arrays and M other variables. At round t , the arrays are denoted by $S_{1,t}[\cdot], S_{2,t}[\cdot], \dots, S_{N,t}[\cdot]$ and the variables by $m_{1,t}, m_{2,t}, \dots, m_{M,t}$. We observe that the output Z_t is of the following form,

$$\begin{aligned} Z_t &= \text{ROT}[\dots \text{ROT}[\text{ROT}[\text{ROT}[V_{1,t}] \otimes \text{ROT}[V_{2,t}]] \\ &\quad \otimes \text{ROT}[V_{3,t}]] \otimes \dots \otimes \text{ROT}[V_{k,t}]] \end{aligned} \quad (6)$$

where $V_{i,t} = m_{g,t}$ or $S_{j,t}[I_l]$; $\text{ROT}[\cdot]$ is the cyclic rotation function either constant or variable depending on the secret state; the function $\otimes[\cdot, \cdot]$ is either *bit-wise XOR* or *addition* modulo 2^n .

Now we describe a general technique to establish a distinguishing attack on an array-based cipher from the above information. We recall that, at the first round (round t_1 in the present context), the internal state is assumed to be uniformly distributed (see Sect. 1.2).

Step 1. [Analogy with *Property 1* of *Example 1*] Observe the elements of the internal state which are involved in the outputs Z_{t_1}, Z_{t_2}, \dots (i.e., the $V_{i,t}$'s in (6)) when the rounds in question are close ($t_1 < t_2 < \dots$).

Step 2. [Bias-inducing state, Analogy with *Property 2* of *Example 1*] Fix a few bits of some array elements (or fix a relation among them) at the initial round t_1 such that *indices* of array-elements in later rounds can be predicted with probability 1 or close to it. More specifically, we search for a *partially specified internal state* such that one or both of the following cases occur due to predictable *index-pointers*.

1. The $V_{i,t}$'s involved in Z_{t_1}, Z_{t_2}, \dots are those array-elements whose bits are already fixed.
2. Each $V_{i,t}$ is dependent on one or more other variables in Z_{t_1}, Z_{t_2}, \dots .

Now, for this case, we compute the bias in the output bits. Below we identify the reasons why an array-based cipher can potentially fall into the above scenarios.

REASON 1. Usually, an array-based cipher uses a number of pseudorandom index-pointers which are updated by the elements of the array. This fact turns out to be a weakness, as fixed values (or a relation) can be assigned to the array-elements such that the index-pointers fetch values from known locations. In other words, the weakness results from the correlation between index-pointers and array-elements which are, although, uniformly distributed individually but not independent of each other.

REASON 2. Barring a few, most of the array-elements do not change in rounds which are close to each other. Therefore, by fixing bits, it is sometimes easy to force the pseudorandom index-pointers fetch certain elements from the arrays in successive rounds.

REASON 3. The size of an index-pointer is small, usually 8 bits irrespective of the size of an array-element which is either 16 bits or 32 bits or 64 bits. Therefore, fixing a small number of bits of the array-elements, it is possible to assign appropriate values to the index-pointers. The less the number of fixed bits, the greater is the bias (note the parameter k' in (5)).

REASON 4. If the rotation operations in the output function are determined by pseudorandom array elements (see (6)) then fixing a few bits of internal state can simplify the function by freeing it from rotation operations. In many cases rotation operations are not present in the function. In any case the output function takes the following form.

$$Z_t = V_{1,t} \circledast V_{2,t} \circledast V_{3,t} \circledast \dots \circledast V_{k,t}.$$

Irrespective of whether ' \circledast ' denotes ' \oplus ' or '+', the following equation holds for the *lsb* of Z_t .

$$Z_{t(0)} = V_{1,t(0)} \oplus V_{2,t(0)} \oplus V_{3,t(0)} \oplus \dots \oplus V_{k,t(0)}.$$

Now by adjusting the index-pointers through fixing bits, if certain equalities among the $V_{i,t}$'s are ensured then $\bigoplus_t Z_{t(0)} = 0$ occurs with probability 1 rather than probability 1/2.

Step 3. [Analogy with *Property 3 of Example 1*] Prove or provide strong evidence that, for the rest of the states other than the bias-inducing state, the bias generated in the previous step is not counterbalanced.

REASON. The internal state of such cipher is huge and uniformly distributed at the initial round. The correlation, detected among the indices and array-elements in Step 2, is fortuitous although not entirely surprising because the variables are not independent. Therefore, the possibility that a bias, produced by an accidental *state*, is *totally* counterbalanced by another accidental state is negligible. In other words, if the bias-inducing state, as explained in Step 2, does not occur, it is likely that at least one of the $V_{i,t}$'s in (6) is uniformly distributed and independent; this fact ensures that the outputs are also uniformly distributed and independent.

Step 4. [Analogy with (5) of *Example 1*] Estimate the overall bias from the results in Step 2 and 3. \square

In the next section, we attack several array-based ciphers following the methods described in this section.

3 Distinguishing Attacks on Array-Based Ciphers or PRBG's

This section describes distinguishing attacks on the ciphers (or PRBG's) Py6, IA, ISAAC, NGG and GGHN – each of which is based on *arrays* and *modular addition*. Due to space constraints, full description of the ciphers is omitted; the reader is kindly referred to the corresponding design papers for details. For each of the ciphers, our task is essentially two-forked as summed up below.

1. **Identification of a Bias-inducing State.** This state is denoted by the event E which adjusts the *index-pointers* in such a way that the *lsb*'s of the outputs are biased. The *lsb*'s of the outputs are potentially vulnerable as they are generated without any carry bits which are nonlinear combinations of input bits (see Step 2 of the general technique described in Sect. 2.2).
2. **Computation of the Probability of Overall Bias.** The probability is calculated considering both E and E^c . As suggested in Step 3 of Sect. 2.2, for each cipher, the *lsb*'s of the outputs are uniformly distributed if the event E does not occur under the assumption mentioned in Sect. 1.2.

Note. For each of the five ciphers attacked in the subsequent sections, it can be shown that, if E (i.e., the bias-inducing state) does not occur then the variable under investigation is uniformly distributed under the assumption of uniformly

distributed *internal state* after the key-setup algorithm. We omit those proofs due to space constraints.

3.1 Bias in the Outputs of Py6

The stream cipher Py6, designed especially for fast software applications by Biham and Seberry in 2005, is one of the modern ciphers that are based on arrays [2,4].² Although the cipher Py, a variant of Py6, was successfully attacked [15,5], Py6 has so far remained alive. The PRBG of Py6 is described in Algorithm 1 (see [2,4] for a detailed discussion).

Algorithm 1. Single Round of Py6

Input: $Y[-3, \dots, 64]$, $P[0, \dots, 63]$, a 32-bit variable s

Output: 64-bit random output

```

/*Update and rotate P*/
1: swap (P[0], P[Y[43]&63]);
2: rotate (P);
/* Update s*/
3: s+ = Y[P[18]] - Y[P[57]];
4: s = ROTL32(s, ((P[26] + 18)&31));
/* Output 8 bytes (least significant byte first)*/
5: output ((ROTL32(s, 25) ⊕ Y[64]) + Y[P[8]]);
6: output ((          s          ⊕ Y[-1]) + Y[P[21]]);
/* Update and rotate Y*/
7: Y[-3] = (ROTL32(s, 14) ⊕ Y[-3]) + Y[P[48]];
8: rotate(Y);

```

Bias-producing State of Py6. Below we identify six conditions among the elements of the S-box P , for which the distribution of $Z_{1,1} \oplus Z_{2,3}$ is biased ($Z_{1,t}$ and $Z_{2,t}$ denote the lower and upper 32 bits of output respectively, at round t).

C1. $P_2[26] \equiv -18 \pmod{32}$; **C2.** $P_3[26] \equiv 7 \pmod{32}$; **C3.** $P_2[18] = P_3[57] + 1$; **C4.** $P_2[57] = P_3[18] + 1$; **C5.** $P_1[8] = 1$; **C6.** $P_3[21] = 62$.

Let the event E denote the simultaneous occurrence of the above conditions ($P[E] \approx 2^{-33.86}$). It can be shown that, if E occurs then $Z_{(0)} = 0$ where Z denotes $Z_{1,1} \oplus Z_{2,3}$ (see the full version of the paper [16]). Now we calculate the probability of occurrence of $Z_{(0)}$.

$$\begin{aligned}
 P[Z_{(0)} = 0] &= P[Z_{(0)} = 0|E] \cdot P[E] + P[Z_{(0)} = 0|E^c] \cdot P[E^c] \\
 &= 1 \cdot 2^{-33.86} + \frac{1}{2} \cdot (1 - 2^{-33.86}) \\
 &= \frac{1}{2} \cdot (1 + 2^{-33.86}).
 \end{aligned} \tag{7}$$

² The cipher has been submitted to the ECRYPT Project [6].

Note that, if Py6 had been an ideal PRBG then the above probability would have been exactly $\frac{1}{2}$.

Remark 2. The above bias can be generalized for rounds t and $t + 2$ ($t > 0$) rather than only rounds 1 and 3.

Remark 3. The main difference between Py and Py6 is that the locations of S-box elements used by one cipher is different from those by the other. The significance of the above results is that it shows that changing the locations of array-elements is futile if the cipher retains some intrinsic weaknesses as explained in Sect. 2.2. Note that Py was attacked with $2^{84.7}$ data while Py6 is with $2^{68.61}$ (explained in Sect. 3.4).

3.2 Biased Outputs in IA and ISAAC

At FSE 1996, R. Jenkins Jr. proposed two fast PRBG's, namely IA and ISAAC, along the lines of the RC4 stream cipher [10]. The round functions of IA and ISAAC are shown in Algorithm 2 and Algorithm 3. Each of them uses an array of 256 elements. The size of an array-element is 16 bits for IA and 32 bits for ISAAC. However, IA and ISAAC can be adapted to work with array-elements of larger size too. For IA, this is the first time that an attack is proposed. For ISAAC, the earlier attack was by Pudovkina who claimed to have deduced its internal state with time $4 \cdot 67 \cdot 10^{1240}$ which was way more than the exhaustive search through the keys of usual size of 256-bit or 128-bit [17]. On the other hand, we shall see later in Sect. 3.4 that our distinguishing attacks can be built with much lower time complexities. The Z_t denotes the output at round t .

Algorithm 2. PRBG of IA

Input: $m[0, 1, \dots, 255]$, 16-bit random variable b

Output: 16-bit random output

- 1: $i = 0$;
 - 2: $x = m[i]$;
 - 3: $m[i] = y = m[\text{ind}(x)] + b \bmod 2^{16}$; /* $\text{ind}(x) = x_{(7,0)}$ */
 - 4: Output = $b = m[\text{ind}(y \gg 8)] + x \bmod 2^{16}$;
 - 5: $i = i + 1 \bmod 256$;
 - 6: Go to step 2;
-

Bias-inducing State of IA. Let $m_t[i_t + 1 \bmod 256] = a$. If the following condition

$$\text{ind}((a + Z_t) \gg 8) = \text{ind}(a) = i_{t+1} \quad (8)$$

is satisfied then

$$Z_{(0)} (= Z_{t(0)} \oplus Z_{t+1(0)}) = 0$$

A pictorial description of the state is provided in the full version of the paper [16]. Let event E occur when (8) holds good. Note that $P[E] = 2^{-16}$ assuming a and Z_t are independent and uniformly distributed. Therefore,

$$\begin{aligned}
 P[Z_{(0)} = 0] &= P[Z_{(0)} = 0|E] \cdot P[E] + P[Z_{(0)} = 0|E^c] \cdot P[E^c] \\
 &= 1 \cdot 2^{-16} + \frac{1}{2} \cdot (1 - 2^{-16}) \\
 &= \frac{1}{2} \cdot (1 + 2^{-16}).
 \end{aligned}
 \tag{9}$$

Algorithm 3. PRBG of ISAAC

Input: $m[0, 1, \dots, 255]$, two 32-bit random variables a and b

Output: 32-bit random output

- 1: $i = 0$;
 - 2: $x = m[i]$;
 - 3: $a = a \oplus (a \lll R) + m[i + K \bmod 256] \bmod 2^{32}$;
 - 4: $m[i + 1] = y = m[ind(x)] + a + b \bmod 2^{32}$; /* $ind(x) = x_{(9,2)}$ */
 - 5: Output = $b = m[ind(y \gg 8)] + x \bmod 2^{32}$;
 - 6: $i = i + 1 \bmod 256$;
 - 7: Go to Step 2.
-

Bias-inducing State of ISAAC. For easy understanding, we rewrite the PRBG of the ISAAC in a simplified manner in Algorithm 3. The variables R and K , described in step 3 of Algorithm 3, depend on the parameter i (see [10] for details); however, we show that our attack can be built independent of those variables.

Let $m_{t-1}[i_t] = x$. Let event E occur when the following equation is satisfied.

$$ind((m_{t-1}[ind(x)] + a_t + b_{t-1}) \gg 8) = i_t.
 \tag{10}$$

If E occurs then $Z_t = x + x \bmod 2^{32}$, i.e., $Z_{t(0)} = 0$ (see the full version of the paper [16]). As a_t , b_{t-1} and x are independent and each of them is uniformly distributed over $\mathbb{Z}_{2^{32}}$, the following equation captures the bias in the output.

$$\begin{aligned}
 P[Z_{t(0)} = 0] &= P[Z_{t(0)} = 0|E] \cdot P[E] + P[Z_{t(0)} = 0|E^c] \cdot P[E^c] \\
 &= 1 \cdot 2^{-8} + \frac{1}{2} \cdot (1 - 2^{-8}) \\
 &= \frac{1}{2} \cdot (1 + 2^{-8}).
 \end{aligned}
 \tag{11}$$

3.3 Biases in the Outputs of NGG and GGHN

Gong *et al.* very recently have proposed two array-based ciphers NGG and GGHN with 32/64-bit word-length [12,8] for very fast software applications. The PRBG's of the ciphers are described in Algorithm 4 and Algorithm 5. Both the ciphers are

claimed to be more than three times as fast as RC4. Due to the introduction of an extra 32-bit random variable k , the GGHN is evidently a stronger version of NGG. We propose attacks on both the ciphers based on the general technique described in Sect. 2.2. Note that the NGG cipher was already experimentally attacked by Wu without theoretical quantification of the attack parameters such as bias, required outputs [19]. For NGG, our attack is new, theoretically justifiable and most importantly, conforms to the basic weaknesses of an array-based cipher, as explained in Sect. 2.2. For GGHN, our attack is the first attack on the cipher. In the following discussion, the Z_t denotes the output at round t .

Algorithm 4. Pseudorandom Bit Generation of NGG

Input: $S[0, 1, \dots, 255]$

Output: 32-bit random output

- 1: $i = 0, j = 0;$
 - 2: $i = i + 1 \bmod 256;$
 - 3: $j = j + S[i] \bmod 256;$
 - 4: Swap ($S[i], S[j]$);
 - 5: Output = $S[S[i] + S[j] \bmod 256];$
 - 6: $S[S[i] + S[j] \bmod 256] = S[i] + S[j] \bmod 2^{32}$
 - 7: Go to step 2;
-

Bias-inducing State of NGG. Let the event E occur, if $i_t = j_t$ and $S_{t+1}[i_{t+1}] + S_{t+1}[j_{t+1}] = 2 \cdot S_t[i_t] \bmod 256$. We observe that, if E occurs then $Z_{t+1(0)} = 0$ (see the full version of the paper [16]). Now we compute $P[Z_{t+1(0)} = 0]$ where $P[E] = 2^{-16}$.

$$\begin{aligned}
 P[Z_{t+1(0)} = 0] &= P[Z_{t+1(0)} = 0|E] \cdot P[E] + P[Z_{t+1(0)} = 0|E^c] \cdot P[E^c] \\
 &= 1 \cdot 2^{-16} + \frac{1}{2} \cdot (1 - 2^{-16}) \\
 &= \frac{1}{2} \cdot (1 + 2^{-16}). \tag{12}
 \end{aligned}$$

Algorithm 5. Pseudorandom Bit Generation of GGHN

Input: $S[0, 1, \dots, 255], k$

Output: 16-bit random output

- 1: $i = 0, j = 0;$
 - 2: $i = i + 1 \bmod 256;$
 - 3: $j = j + S[i] \bmod 256;$
 - 4: $k = k + S[j] \bmod 2^{32};$
 - 5: Output = $S[S[i] + S[j] \bmod 256] + k \bmod 2^{32};$
 - 6: $S[S[i] + S[j] \bmod 256] = k + S[i] \bmod 2^{32};$
 - 7: Go to step 2;
-

Bias-producing State of GGHN. If $S_t[i_t] = S_{t+1}[j_{t+1}]$ and $S_t[j_t] = S_{t+1}[i_{t+1}]$ (denote it by event E) then $Z_{t+1(0)} = 0$ (see the full version of the paper [16]). Now we compute $P[Z_{t+1(0)} = 0]$ where $P[E] = 2^{-16}$.

$$\begin{aligned}
P[Z_{t+1(0)} = 0] &= P[Z_{t+1(0)} = 0|E] \cdot P[E] + P[Z_{t+1(0)} = 0|E^c] \cdot P[E^c] \\
&= 1 \cdot 2^{-16} + \frac{1}{2} \cdot (1 - 2^{-16}) \\
&= \frac{1}{2} \cdot (1 + 2^{-16}).
\end{aligned} \tag{13}$$

3.4 Data and Time of the Distinguishing Attacks

In the section we compute the data and time complexities of the distinguishers derived from the biases computed in the previous sections. A *distinguisher* is an algorithm which distinguishes a stream of bits from a perfectly random stream of bits, that is, a stream of bits that has been chosen according to the uniform distribution. The *advantage* of a distinguisher is the measure of its success rate (see [1] for a detailed discussion).

Let there be n binary random variables z_1, z_2, \dots, z_n which are independent of each other and each of them follows the distribution D_{BIAS} . Let the uniform distribution on alphabet \mathbb{Z}_2 be denoted by D_{UNI} . Method to construct an *optimal distinguisher* with a fixed number of samples is given in [1].³ While the detailed description of an *optimal distinguisher* is omitted, the following theorem determines the number of samples required by an *optimal distinguisher* to attain an advantage of 0.5 which is considered a reasonable goal.

Theorem 1. *Let the input to an optimal distinguisher be a realization of the binary random variables $z_1, z_2, z_3, \dots, z_n$ where each z_i follows D_{BIAS} . To attain an advantage of more than 0.5, the least number of samples required by the optimal distinguisher is given by the following formula*

$$n = 0.4624 \cdot M^2 \quad \text{where}$$

$$P_{D_{\text{BIAS}}}[z_i = 0] - P_{D_{\text{UNI}}}[z_i = 0] = \frac{1}{M}.$$

Proof. See Sect. 5 of [15] for the proof.

Now D_{UNI} is known and D_{BIAS} can be determined from (7) for Py6, (9) for IA, (11) for ISAAC, (12) for NGG, (13) for GGHN. In Table 2, we list the data and time complexities of the distinguishers. Our experiments agree well with the theoretical results. The constant in $O(m)$ is determined by the time taken by single round of the corresponding cipher.

3.5 A Note on IBAA, Pypy and HC-256

IBAA, Pypy and HC-256 are the array-oriented ciphers/PRBG's which are still free from any attacks. The IBAA works in a similar way as the ISAAC works,

³ Given a fixed number of samples, an *optimal distinguisher* attains the maximum advantage.

Table 2. Data and time of the distinguishers with advantage exceeding 0.5

PRBG	M	Bytes of a single stream = $0.4624 \cdot M^2$	Time
Py6	$2^{34.86}$	$2^{68.61}$	$O(2^{68.61})$
IA	2^{17}	$2^{32.89}$	$O(2^{32.89})$
ISAAC	2^9	$2^{16.89}$	$O(2^{16.89})$
NGG	2^{17}	$2^{32.89}$	$O(2^{32.89})$
GGHN	2^{17}	$2^{32.89}$	$O(2^{32.89})$

except for the variable a which plays an important role in the output generation of IBAA [10]. It seems that a relation has to be discovered among the values of the parameter a at different rounds to successfully attack IBAA. Pypy is a slower variant of Py and Py6 [3]. Pypy produces 32 bits per round when each of Py and Py6 produces 64 bits. To attack Pypy a relation need to be found among the elements which are separated by at least three rounds. To attack HC-256 [18], some correlations need to be known among the elements which are cyclically rotated by constant number of bits.

4 Conclusion

In this paper, we have studied array-based stream ciphers or PRBG's in a general framework to assess their resistance against certain distinguishing attacks originating from the correlation between index-pointers and array-elements. We show that the weakness becomes more profound because of the usage of simple modular additions in the output generation function. In the unified framework we have attacked five modern array-based stream ciphers (or PRBG's) Py6, IA, ISAAC, NGG, GGHN with data complexities $2^{68.61}$, $2^{32.89}$, $2^{16.89}$, $2^{32.89}$ and $2^{32.89}$ respectively. We also note that some other array-based stream ciphers (or PRBG's) IBAA, Pypy, HC-256 still do not come under any threats, however, the algorithms need to be analyzed more carefully in order to be considered secure. We believe that our investigation will throw light on the security of array-based stream ciphers in general and can possibly be extended to analyze other types of ciphers too.

Acknowledgments

We thank Jongsung Kim, Hongjun Wu, Gautham Sekar for useful discussions. We also acknowledge the constructive comments of the anonymous reviewers of Asiacypt 2006.

References

1. T. Baignères, P. Junod and S. Vaudenay, "How Far Can We Go Beyond Linear Cryptanalysis?," *Asiacypt 2004* (P. Lee, ed.), vol. 3329 of *LNCS*, pp. 432–450, Springer-Verlag, 2004.
2. E. Biham, J. Seberry, "Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays," eSTREAM, ECRYPT Stream Cipher Project, Report 2005/023, 2005.

3. Eli Biham and Jennifer Seberry, "Pypy: Another Version of Py," eSTREAM, ECRYPT Stream Cipher Project, Report 2006/038, 2006.
4. E. Biham, J. Seberry, "C Code of Py6," as available from <http://www.ecrypt.eu.org/stream/py.html>, eSTREAM, ECRYPT Stream Cipher Project, 2005.
5. P. Crowley, "Improved Cryptanalysis of Py," *Workshop Record of SASC 2006 – Stream Ciphers Revisited*, ECRYPT Network of Excellence in Cryptology, February 2006, Leuven (Belgium), pp. 52–60.
6. Ecrypt, <http://www.ecrypt.eu.org>.
7. S. Fluhrer, D. McGrew, "Statistical Analysis of the Alleged RC4 Keystream Generator," *Fast Software Encryption 2000* (B. Schneier, ed.), vol. 1978 of *LNCS*, pp. 19–30, Springer-Verlag, 2000.
8. G. Gong, K. C. Gupta, M. Hell, Y. Nawaz, "Towards a General RC4-Like Keystream Generator," *First SKLOIS Conference, CISC 2005* (D. Feng, D. Lin, M. Yung, eds.), vol. 3822 of *LNCS*, pp. 162–174, Springer-Verlag, 2005.
9. S. Halevi, D. Coppersmith, C. S. Jutla, "Scream: A Software-Efficient Stream Cipher," *Fast Software Encryption 2002* (J. Daemen and V. Rijmen, eds.), vol. 2365 of *LNCS*, pp. 195–209, Springer-Verlag, 2002.
10. Robert J. Jenkins Jr., "ISAAC," *Fast Software Encryption 1996* (D. Gollmann, ed.), vol. 1039 of *LNCS*, pp. 41–49, Springer-Verlag, 1996.
11. I. Mantin, A. Shamir, "A Practical Attack on Broadcast RC4," *Fast Software Encryption 2001* (M. Matsui, ed.), vol. 2355 of *LNCS*, pp. 152–164, Springer-Verlag, 2001.
12. Y. Nawaz, K. C. Gupta, and G. Gong, "A 32-bit RC4-like Keystream Generator," Cryptology ePrint Archive, 2005/175.
13. NESSIE: New European Schemes for Signature, Integrity and Encryption, <http://www.cryptoneessie.org>.
14. Souradyuti Paul, Bart Preneel, "A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher," *Fast Software Encryption 2004* (B. Roy, ed.), vol. 3017 of *LNCS*, pp. 245–259, Springer-Verlag, 2004.
15. Souradyuti Paul, Bart Preneel, Gautham Sekar, "Distinguishing Attacks on the Stream Cipher Py," *Fast Software Encryption 2006* (M. Robshaw, ed.), vol. 4047 of *LNCS*, Springer-Verlag, pp. 405–421, 2006 (to appear).
16. Souradyuti Paul, Bart Preneel, "On the (In)security of Stream Ciphers Based on Arrays and Modular Addition (Full Version)," Cryptology ePrint Archive: Report 2005/448, IACR, 2005, Available online at <http://eprint.iacr.org/2005/448>.
17. Marina Pudovkina, "A known plaintext attack on the ISAAC keystream generator," *Cryptology ePrint Archive: Report 2001/049*, IACR, 2001.
18. H. Wu, "A New Stream Cipher HC-256," *Fast Software Encryption 2004* (B. Roy, ed.), vol. 3017 of *LNCS*, pp. 226–244, Springer-Verlag, 2004.
19. H. Wu, "Cryptanalysis of a 32-bit RC4-like Stream Cipher," Cryptology ePrint Archive, 2005/219.
20. Bartosz Zoltak, "VMPC One-Way Function and Stream Cipher," *Fast Software Encryption 2004* (B. Roy, ed.), vol. 3017 of *LNCS*, pp. 210–225, Springer-Verlag, 2004.

Construction and Analysis of Boolean Functions of $2t + 1$ Variables with Maximum Algebraic Immunity*

Na Li and Wen-Feng Qi

Department of Applied Mathematics, Zhengzhou Information Engineering University, Zhengzhou, 450002, China
mylina_1980@yahoo.com.cn, wenfeng.qi@263.net

Abstract. In this paper, we study the construction of $(2t + 1)$ -variable Boolean functions with maximum algebraic immunity, and we also analyze some other cryptographic properties of this kind of functions, such as nonlinearity, resilience. We first identify several classes of this kind of functions. Further, some necessary conditions of this kind of functions which also have higher nonlinearity are obtained. In this way, a modified construction method is proposed to possibly obtain $(2t + 1)$ -variable Boolean functions which have maximum algebraic immunity and higher nonlinearity, and a class of such functions is also obtained. Finally, we present a sufficient and necessary condition of $(2t + 1)$ -variable Boolean functions with maximum algebraic immunity which are also 1-resilient.

Keywords: Algebraic attack, algebraic immunity, Boolean functions, balancedness, nonlinearity, resilience.

1 Introduction

The recent progress in research related to algebraic attacks [1,2,5,6] seems to threaten all LFSR-based stream ciphers. It is known that Boolean functions used in stream ciphers should have high algebraic degree [11]. However, a Boolean function may have low degree multiples even if its algebraic degree is high. By this fact it is possible to obtain an over-defined system of multivariate equations of low degree whose unknowns are the bits of the initialization of the LFSR(s). Then the secret key can be discovered by solving the system.

To measure the resistance to algebraic attacks, a new cryptographic property of Boolean functions called algebraic immunity (AI) has been proposed by W. Meier *et al.* [16]. When used in a cryptosystem, a Boolean function should have high AI. Now, it is known that the AI of an n -variable Boolean function is upper bounded by $\lceil \frac{n}{2} \rceil$ [6,16]. Balancedness, nonlinearity and correlation-immunity are three other important cryptographic criteria. In some sense, algebraic immunity

* This work was supported by National Nature Science Foundation of China under Grant number 60373092.

is compatible with the former two criteria: a Boolean functions with low nonlinearity will have low AI [7,14], a Boolean function of an odd number of variables with maximum AI must be balanced [7]. The existence of links between algebraic immunity and correlation-immunity remains open.

Constructions of Boolean functions with maximum AI are obviously important. Further, it is more important to construct these functions which also satisfy some other criteria (such as balancedness, a high nonlinearity, a high correlation-immunity order, ...). Some classes of symmetric Boolean functions with maximum AI were obtained in [3] and [9], and it was shown in [12] that there is only one such symmetric function (besides its complement) when the number of input variables is odd. A construction keeping in mind the basic theory of algebraic immunity was presented in [9], which also provided some functions with maximum AI. In [4], Carlet introduced a general method (for any number of variables) and an algorithm (for an even number of variables) for constructing balanced functions with maximum AI. In [13], a method was proposed for constructing functions of an odd number of variables with maximum AI, which convert the problem of constructing such a function to the problem of finding an invertible submatrix of a $2^{n-1} \times 2^{n-1}$ matrix. And it was stated that any such function can be obtained by this method.

In this paper, we study the construction of $(2t + 1)$ -variable Boolean functions with maximum AI, and we also analyze some other cryptographic properties of this kind of functions. From the characteristic of the matrix used in the construction proposed in [13], we obtain some necessary or sufficient conditions of $(2t + 1)$ -variable Boolean functions with maximum AI. Further, by studying the Walsh spectra of this kind of functions, we obtain some necessary conditions of this kind of functions which also have higher nonlinearity and thus we propose a modified construction to obtain such functions. We finally present a sufficient and necessary condition of $(2t + 1)$ -variable Boolean functions with maximum AI which are also 1-resilient.

2 Preliminaries

Let \mathbb{F}_2^n be the set of all n -tuples of elements in the finite field \mathbb{F}_2 . To avoid confusion with the usual sum, we denote the sum over \mathbb{F}_2 by \oplus .

A Boolean function of n variables is a function from \mathbb{F}_2^n into \mathbb{F}_2 . Any n -variable Boolean function f can be uniquely expressed by a polynomial in $\mathbb{F}_2[x_1, \dots, x_n]/(x_1^2 - x_1, \dots, x_n^2 - x_n)$, which is called its algebraic normal form (ANF). The algebraic degree of f , denoted by $\deg(f)$, is the degree of this polynomial. Boolean function f can also be identified by a binary string of length 2^n , called its truth table, which is defined as

$$(f(0, 0, \dots, 0), f(1, 0, \dots, 0), f(0, 1, \dots, 0), \dots, f(1, 1, \dots, 1)).$$

Let

$$1_f = \{X \in \mathbb{F}_2^n | f(X) = 1\}, 0_f = \{X \in \mathbb{F}_2^n | f(X) = 0\}.$$

The set 1_f (resp. 0_f) is called the on set (resp. off set). The cardinality of 1_f , denoted by $wt(f)$, is called the Hamming weight of f . We say that an n -variable Boolean function f is balanced if $wt(f) = 2^{n-1}$. The Hamming distance between two functions f and g , denoted by $d(f, g)$, is the Hamming weight of $f \oplus g$. Let $S = (s_1, s_2, \dots, s_n) \in \mathbb{F}_2^n$, the Hamming weight of S , denoted by $wt(S)$, is the number of 1's in $\{s_1, s_2, \dots, s_n\}$.

Walsh spectra is an important tool for studying Boolean functions. Let $X = (x_1, \dots, x_n)$ and $S = (s_1, \dots, s_n)$ both belonging to \mathbb{F}_2^n and their inner product $X \cdot S = x_1s_1 \oplus \dots \oplus x_ns_n$. Let f be a Boolean function of n variables. Then the Walsh transform of f is an integer valued function over \mathbb{F}_2^n which is defined as

$$W_f(S) = \sum_{X \in \mathbb{F}_2^n} (-1)^{f(X) \oplus X \cdot S}.$$

Affine functions are those Boolean functions of degree at most 1. The nonlinearity of an n -variable Boolean function f is its Hamming distance from the set of all n -variable affine functions, i.e.,

$$nl(f) = \min\{d(f, g) | g \text{ is an affine function}\}.$$

The nonlinearity of f can be described by its Walsh spectra as $nl(f) = 2^{n-1} - \frac{1}{2} \max_{S \in \mathbb{F}_2^n} |W_f(S)|$. Correlation immune functions and resilient functions are two important classes of Boolean functions. A function is m th order correlation immune (resp. m -resilient) if and only if its Walsh spectra satisfies

$$W_f(S) = 0, \text{ for } 1 \leq wt(S) \leq m \text{ (resp. } 0 \leq wt(S) \leq m).$$

Definition 1. [16] For a given n -variable Boolean function f , a nonzero n -variable Boolean function g is called an annihilator of f if $f \cdot g = 0$, and the algebraic immunity of f , denoted by $AI(f)$, is the minimum value of d such that f or $f \oplus 1$ admits an annihilating function of degree d .

For convenience, two orderings on vectors and monomials are defined as follows.

Definition 2. A vector ordering $<_v$ on \mathbb{F}_2^n is defined as:

let $(a_1, \dots, a_n), (b_1, \dots, b_n) \in \mathbb{F}_2^n$, then $(a_1, \dots, a_n) <_v (b_1, \dots, b_n)$ if and only if $\sum_{i=1}^n a_i < \sum_{i=1}^n b_i$, or $\sum_{i=1}^n a_i = \sum_{i=1}^n b_i$ and there exists $1 \leq i < n$ such that $a_i > b_i, a_j = b_j$ for $1 \leq j < i$.

Example 1. If $n = 3$, then $(0, 0, 0) <_v (1, 0, 0) <_v (0, 1, 0) <_v (0, 0, 1) <_v (1, 1, 0) <_v (1, 0, 1) <_v (0, 1, 1) <_v (1, 1, 1)$.

Definition 3. A monomial ordering $<_m$ on $\mathbb{F}_2[x_1, \dots, x_n]/(x_1^2 - x_1, \dots, x_n^2 - x_n)$ is defined as:

let $x_1^{a_1} \dots x_n^{a_n}, x_1^{b_1} \dots x_n^{b_n} \in \mathbb{F}_2[x_1, \dots, x_n]/(x_1^2 - x_1, \dots, x_n^2 - x_n)$, then $x_1^{a_1} \dots x_n^{a_n} <_m x_1^{b_1} \dots x_n^{b_n}$ if and only if $(a_1, \dots, a_n) <_v (b_1, \dots, b_n)$.

It is clear that $<_v$ and $<_m$ are both total orderings.

Let A be an $l \times l$ matrix, and integers $1 \leq i_1, i_2, \dots, i_k \leq l, 1 \leq j_1, j_2, \dots, j_k \leq l$. Denoted by $A_{(i_1, \dots, i_k)}$ the $k \times l$ matrix with the r th ($1 \leq r \leq k$) row vector equal to the i_r th row vector of A , and $A_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ the $k \times k$ matrix with the r th ($1 \leq r \leq k$) column vector equal to the j_r th column vector of $A_{(i_1, \dots, i_k)}$.

3 Construction of Boolean Functions with Maximum AI

In this section, we briefly review the method to construct Boolean functions with maximum AI proposed in [13].

Let n be a positive integer, $X = (x_1, \dots, x_n) \in \mathbb{F}_2^n$. Let

$$v(X) = (1, x_1, \dots, x_n, x_1x_2, \dots, x_{n-1}x_n, \dots, \\ x_1 \cdots x_{\lfloor \frac{n}{2} \rfloor - 1}, \dots, x_{\lfloor \frac{n}{2} \rfloor + 2} \cdots x_n) \in \mathbb{F}_2^{\sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 1} \binom{n}{i}},$$

where the monomials are ordered according to the ordering $<_m$. It is clear that $\sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 1} \binom{n}{i} = 2^{n-1}$ when n is odd. Let f be an n -variable Boolean function, let $V(1_f)$ denote the $wt(f) \times \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 1} \binom{n}{i}$ matrix with the set of row vectors $\{v(X) | X \in 1_f\}$, and $V(0_f)$ denote the $(2^n - wt(f)) \times \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 1} \binom{n}{i}$ matrix with the set of row vectors $\{v(X) | X \in 0_f\}$.

Lemma 1. [3,9] *Let odd $n = 2t + 1$ and f be an n -variable Boolean function which satisfies*

$$f(X) = \begin{cases} a & \text{for } wt(X) \leq t \\ a \oplus 1 & \text{for } wt(X) > t \end{cases},$$

where $a \in \mathbb{F}_2$, then $AI(f) = t + 1$.

When $a = 1$, the function described in Lemma 1 is called the majority function, and we denote it by F_n . It is clear that F_n is balanced. We arrange the vectors in 1_{F_n} (resp. 0_{F_n}) according to the order $<_v$, and denote them by $X_1, \dots, X_{2^{n-1}}$ (resp. $Y_1, \dots, Y_{2^{n-1}}$), i.e. $X_1 <_v \dots <_v X_{2^{n-1}}$ (resp. $Y_1 <_v \dots <_v Y_{2^{n-1}}$). Let $X_j = (x_{j,1}, \dots, x_{j,n})$ (resp. $Y_i = (y_{i,1}, \dots, y_{i,n})$). The i th row vector of $V(1_{F_n})$ (resp. $V(0_{F_n})$) is $v(X_i)$ (resp. $v(Y_i)$).

The idea of the construction proposed in [13] is to obtain a new function by changing the values of the majority function at some vectors. The problem of finding out the appropriate vectors is converted to the problem of finding out a $k \times k$ invertible submatrix of the $2^{n-1} \times 2^{n-1}$ invertible matrix $W = V(0_{F_n})V(1_{F_n})^{-1}$.

Theorem 1. [13] *Let $n = 2t + 1$, and f an n -variable Boolean function. Then, $AI(f) = t + 1$ if and only if there exist integers $1 \leq i_1 < \dots < i_k \leq 2^{n-1}, 1 \leq j_1 < \dots < j_k \leq 2^{n-1}$, such that $f = f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ and $W_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is invertible, where $f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is defined as*

$$f_{(i_1, \dots, i_k; j_1, \dots, j_k)}(X) = \begin{cases} F_n(X) \oplus 1 & \text{if } X \in \{X_{j_1}, \dots, X_{j_k}, Y_{i_1}, \dots, Y_{i_k}\} \\ F_n(X) & \text{else} \end{cases}. \quad (1)$$

Construction 1. [13] Let $n = 2t + 1$. The following method can generate a Boolean function of n variables with maximum AI.

Step1: Select randomly an integer $1 \leq k \leq 2^{n-2}$ and k integers $1 \leq i_1 < \dots < i_k \leq 2^{n-1}$.

Step2: Find out k integers $1 \leq j_1 < \dots < j_k \leq 2^{n-1}$, such that the j_1 th, ..., j_k th column vectors of $W_{(i_1, \dots, i_k)}$ are linearly independent.

Then, the Boolean function $f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ defined by (1) has AI $t + 1$.

Remark 1. 1) For any fixed $1 \leq k \leq 2^{n-2}$ and any k integers $1 \leq i_1 < \dots < i_k \leq 2^{n-1}$, there always exist k integers $1 \leq j_1 < \dots < j_k \leq 2^{n-1}$ such that $W_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is invertible.

2) Any Boolean function of $2t + 1$ variables with maximum AI can be constructed by this method.

For the rest of this paper, we always suppose $n = 2t + 1$.

4 Properties of W and Several Classes of n -Variable Boolean Functions with Maximum AI

In this section, we first show some important properties of the matrix $W = V(0_{F_n})V(1_{F_n})^{-1}$, then use these conclusions to obtain some necessary or sufficient conditions of n -variable Boolean function achieving maximum AI.

Let A be a $2^{n-1} \times 2^{n-1}$ matrix, and divide A into $(t+1)^2$ submatrixes, denoted by $A_{i,j}$, $1 \leq i \leq t + 1$, $1 \leq j \leq t + 1$, defined as

$$A_{i,j} = A_{(r_{i-1}+1, r_{i-1}+2, \dots, r_i; s_{j-1}+1, s_{j-1}+2, \dots, s_j)},$$

where

$$r_l = \begin{cases} 0 & \text{if } l = 0 \\ \sum_{k=1}^l \binom{n}{t+k} & \text{if } l > 0 \end{cases}, s_l = \begin{cases} 0 & \text{if } l = 0 \\ \sum_{k=0}^{l-1} \binom{n}{k} & \text{if } l > 0 \end{cases}.$$

It is clear that the row (resp. column) vectors of $W_{i,j}$ correspond to the vectors in \mathbb{F}_2^n with Hamming weight $i + t$ (resp. $j - 1$).

Proposition 1. [10] $V(1_{F_n})^{-1} = V(1_{F_n})$.

Proposition 2. Let $W = V(0_{F_n})V(1_{F_n})^{-1}$, then

$$W_{i,j} = \begin{cases} \mathbf{0} & \text{if } \bigoplus_{r=0}^{t-j+1} \binom{t+i-j+1}{r} = 0 \\ V(0_{F_n})_{i,j} & \text{else} \end{cases}, \text{ for } 1 \leq i, j \leq t + 1,$$

where $\mathbf{0}$ denotes the matrix with all entries 0.

Proof. By Proposition 1, $W = V(0_{F_n})V(1_{F_n})^{-1} = V(0_{F_n})V(1_{F_n})$. Let $Y = (y_1, \dots, y_n) \in 0_{F_n}$ and $wt(Y) = i > t$, $x_{r_1} \cdots x_{r_j}$ be a monomial of degree j ($0 \leq j \leq t$). Denote the transpose of the column vector of $V(1_{F_n})$ corresponding to

$x_{r_1} \cdots x_{r_j}$ by $u(x_{r_1} \cdots x_{r_j})$. That is, $u(x_{r_1} \cdots x_{r_j})$ is the evaluation of $x_{r_1} \cdots x_{r_j}$ at the vectors belonging to 1_{F_n} . We can represent $u(x_{r_1} \cdots x_{r_j})$ as

$$\begin{aligned} & (g(1), g(x_1), \dots, g(x_n), g(x_1x_2), g(x_1x_3), \dots, \\ & g(x_{n-1}x_n), \dots, g(x_1 \cdots x_t), \dots, g(x_{t+2} \cdots x_n)), \end{aligned} \quad (2)$$

where g is a function on the monomials of degree at most t , which satisfies

$$g(x_1^{a_1} \cdots x_n^{a_n}) = \begin{cases} 1 & \text{if } x_{r_1} \cdots x_{r_j} | x_1^{a_1} \cdots x_n^{a_n} \\ 0 & \text{else} \end{cases}. \quad (3)$$

On the other hand, we can also represent $v(Y)$ as

$$\begin{aligned} & (h(1), h(x_1), \dots, h(x_n), h(x_1x_2), h(x_1x_3), \dots, \\ & h(x_{n-1}x_n), \dots, h(x_1 \cdots x_t), \dots, h(x_{t+2} \cdots x_n)), \end{aligned} \quad (4)$$

where h is a function on the monomials of degree at most t , which satisfies

$$h(x_1^{a_1} \cdots x_n^{a_n}) = \begin{cases} 1 & \text{if } x_1^{a_1} \cdots x_n^{a_n} | x_1^{y_1} \cdots x_n^{y_n} \\ 0 & \text{else} \end{cases}. \quad (5)$$

Denote the inner product of $v(Y)$ and $u(x_{r_1} \cdots x_{r_j})$ by c .

If y_{r_1}, \dots, y_{r_j} are not all 1, by (2), (3), (4) and (5), we have $c = 0 = h(x_{r_1} \cdots x_{r_j})$. If y_{r_1}, \dots, y_{r_j} are all 1, we have $h(x_{r_1} \cdots x_{r_j}) = 1$ and

$$c = \bigoplus_{\substack{x_{r_1} \cdots x_{r_j} | x_1^{a_1} \cdots x_n^{a_n}, \\ x_1^{a_1} \cdots x_n^{a_n} | x_1^{y_1} \cdots x_n^{y_n} \\ wt(a_1, \dots, a_n) \leq t}} 1 = \bigoplus_{r=0}^{t-j} \binom{i-j}{r}.$$

It is clear that the row (resp. column) vectors of $W_{i,j}$ correspond to the vectors in \mathbb{F}_2^n with Hamming weight $i+t$ (resp. $j-1$). Therefore, we complete the proof.

Corollary 1. 1) For any $2 \leq i \leq t+1$, $W_{i,t+2-i} = \mathbf{0}$.

2) For any $1 \leq j \leq t+1$, $W_{1,j} = V(0_{F_n})_{1,j}$.

3) For any $1 \leq i \leq t+1$, $W_{i,t+1} = V(0_{F_n})_{i,t+1}$.

Proof. 1) If $2 \leq i \leq t+1$ and $j = t+2-i$, then

$$\bigoplus_{r=0}^{t-j+1} \binom{t+i-j+1}{r} = \bigoplus_{r=0}^{i-1} \binom{2i-1}{r} = 2^{2i-2} \bmod 2 = 0.$$

2) If $i = 1$, then

$$\bigoplus_{r=0}^{t-j+1} \binom{t+i-j+1}{r} = \bigoplus_{r=0}^{t-j+1} \binom{t-j+2}{r} = 2^{t-j+2} - 1 \bmod 2 = 1.$$

3) If $j = t + 1$, then

$$\bigoplus_{r=0}^{t-j+1} \binom{t+i-j+1}{r} = 1.$$

We can obtain some necessary conditions of n -variable Boolean functions with maximum AI.

Theorem 2. *Let $1 \leq k \leq 2^{n-1}$, $1 \leq i_1 < \dots < i_k \leq 2^{n-1}$, $1 \leq j_1 < \dots < j_k \leq 2^{n-1}$. If there exist $0 \leq j \leq t$, $t + 1 \leq i \leq n$ such that $\bigoplus_{r=0}^{t-j} \binom{i-j}{r} = 0$, and*

$$\#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) = j\} + \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) = i\} > k,$$

then, $AI(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) < t + 1$.

Proof. By Theorem 1, it is sufficient to show that $W_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is not invertible. By Proposition 2 and the first condition, we have that $W_{i-t, j+1} = \mathbf{0}$. Then the second condition implies that $W_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ has a submatrix with the number of rows and columns greater than k whose entries are all 0. Therefore, $W_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is not invertible.

Corollary 2. *Let $1 \leq k \leq 2^{n-1}$, $1 \leq i_1 < \dots < i_k \leq 2^{n-1}$, $1 \leq j_1 < \dots < j_k \leq 2^{n-1}$. If there exists $0 \leq r \leq t - 1$ such that*

$$\#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) = r\} + \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) = n - r\} > k,$$

then, $AI(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) < t + 1$.

In the following of this section, several classes of n -variable Boolean functions with maximum AI are provided.

Theorem 3. *Let $1 \leq k \leq 2^{n-1}$, $1 \leq i_1 < \dots < i_k \leq 2^{n-1}$, $1 \leq j_1 < \dots < j_k \leq 2^{n-1}$. If the following conditions are both satisfied, then $AI(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) = t + 1$.*

1) *There exist $1 \leq a_1 < \dots < a_s \leq n$, such that $x_{j_r, a_1} = \dots = x_{j_r, a_s} = 0$ for $1 \leq r \leq k$.*

2) *For any X_{j_r} ($1 \leq r \leq k$), there exists correspondingly $Y_{i_r'} \in \{Y_{i_1}, \dots, Y_{i_k}\}$, such that $y_{i_r', a} = x_{j_r, a}$ for $a \notin \{a_1, \dots, a_s\}$, and*

$$\bigoplus_{l=0}^{t-wt(X_{j_r})} \binom{wt(Y_{i_r'}) - wt(X_{j_r})}{l} = 1.$$

Proof. If X_{j_1}, \dots, X_{j_k} and Y_{i_1}, \dots, Y_{i_k} satisfy the two conditions, then by Proposition 2, $W_{(i_1', \dots, i_k'; j_1, \dots, j_k)}$ is in the form of lower triangular with all entries on the diagonal equal to 1. Therefore $W_{(i_1', \dots, i_k'; j_1, \dots, j_k)}$ is invertible, which implies that $W_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is invertible, and the result holds by Theorem 1.

Example 2. Let $n=7$, $L_1 = \{(1, 0, 0, 0, 0, 0, 0), (0, 1, 1, 0, 0, 0, 0), (0, 0, 1, 1, 0, 0, 0), (1, 1, 1, 0, 0, 0, 0)\} \subseteq 1_{F_n}$, $L_2 = \{(1, 0, 0, 0, 1, 1, 1), (0, 1, 1, 0, 1, 1, 0), (0, 0, 1, 1, 0, 1, 1), (1, 1, 1, 0, 1, 1, 1)\} \subseteq 0_{F_n}$. Then the function

$$f(X) = \begin{cases} F_n(X) \oplus 1 & \text{if } X \in L_1 \cup L_2 \\ F_n(X) & \text{else} \end{cases}$$

has AI 4.

Theorem 4. Let $1 \leq 2k \leq 2^{n-1}$, $1 \leq i_1 < \dots < i_{2k} \leq 2^{n-1}$, $1 \leq j_1 < \dots < j_{2k} \leq 2^{n-1}$. $wt(X_{j_r}) = w_1$, $wt(Y_{i_r}) = w'_1$ for $1 \leq r \leq k$, and $wt(X_{j_r}) = w_2$, $wt(Y_{i_r}) = w'_2$ for $k+1 \leq r \leq 2k$. If one of the following two conditions is satisfied, then $AI(f_{(i_1, \dots, i_{2k}; j_1, \dots, j_{2k})}) = t+1$.

- 1) $\bigoplus_{r=0}^{t-w_1} \binom{w'_1-w_1}{r}$ and $\bigoplus_{r=0}^{t-w_2} \binom{w'_1-w_2}{r}$ are not both 1, and

$$AI(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) = AI(f_{(i_{k+1}, \dots, i_{2k}; j_{k+1}, \dots, j_{2k})}) = t+1.$$

- 2) $\bigoplus_{r=0}^{t-w_1} \binom{w'_1-w_1}{r}$ and $\bigoplus_{r=0}^{t-w_2} \binom{w'_2-w_2}{r}$ are not both 1, and

$$AI(f_{(i_1, \dots, i_k; j_{k+1}, \dots, j_{2k})}) = AI(f_{(i_{k+1}, \dots, i_{2k}; j_1, \dots, j_k)}) = t+1.$$

Proof. Let M denote the $2k \times 2k$ matrix $W_{(i_1, \dots, i_{2k}; j_1, \dots, j_{2k})}$. The first condition implies that $M_{(1, \dots, k; 1, \dots, k)}$ and $M_{(k+1, \dots, 2k; k+1, \dots, 2k)}$ are both invertible, and at least one of $M_{(1, \dots, k; k+1, \dots, 2k)}$ and $M_{(k+1, \dots, 2k; 1, \dots, k)}$ is $\mathbf{0}$. Then, M is invertible, and the result holds by Theorem 1.

If the second condition is satisfied, the result can be proved in the same way.

Example 3. Let $n = 7$, $L_1 = \{(0, 0, 0, 0, 1, 1, 0), (0, 0, 0, 0, 1, 0, 1), (0, 0, 0, 0, 0, 1, 1), (1, 1, 0, 0, 1, 0, 0), (1, 1, 0, 0, 0, 1, 0), (1, 1, 0, 0, 0, 0, 1)\}$, $L_2 = \{(1, 1, 0, 0, 1, 1, 0), (1, 1, 0, 0, 1, 0, 1), (1, 1, 0, 0, 0, 1, 1), (1, 1, 1, 1, 1, 0, 0), (1, 1, 1, 1, 0, 1, 0), (1, 1, 1, 1, 0, 0, 1)\}$. Then the function

$$f(X) = \begin{cases} F_n(X) \oplus 1 & \text{if } X \in L_1 \cup L_2 \\ F_n(X) & \text{else} \end{cases}$$

has AI 4.

Theorem 5. Let $1 \leq k \leq n$, Y_{i_1}, \dots, Y_{i_k} belong to 0_{F_n} and their Hamming weight are w_1, \dots, w_k , respectively. If

- 1) $\bigoplus_{r=0}^{t-1} \binom{w_i-1}{r} = 1$ for $1 \leq i \leq k$, and

- 2) there exist $1 \leq j_1 < \dots < j_k \leq n$, such that the j_1 th, \dots , j_k th column of

the matrix $\begin{pmatrix} Y_{i_1} \\ \dots \\ Y_{i_k} \end{pmatrix}$ are linearly independent,

then, $AI(f_{(i_1, \dots, i_k; j_1+1, \dots, j_k+1)}) = t+1$.

Proof. By Proposition 2, $W_{(i_1, \dots, i_k; j_1+1, \dots, j_k+1)}$ is invertible if the two conditions are both satisfied, and the result holds by Theorem 1.

Example 4. Let $n=7$, $L_1=\{(1, 0, 0, 0, 0, 0, 0), (0, 1, 0, 0, 0, 0, 0), (0, 0, 1, 0, 0, 0, 0)\}$, $L_2=\{(1, 0, 1, 0, 1, 1, 1), (0, 1, 1, 0, 1, 0, 1), (1, 1, 1, 1, 0, 1, 0)\}$. Then the function

$$f(X) = \begin{cases} F_n(X) \oplus 1 & \text{if } X \in L_1 \cup L_2 \\ F_n(X) & \text{else} \end{cases}$$

has AI 4.

5 Nonlinearity and Resilience of Boolean Functions with Maximum AI

At first, we give the Walsh spectra of majority functions. Note that although the first item and the case of $wt(S) = 1$ in the second item in the following lemma have been given in [9], we still give the proof for completeness.

Lemma 2. *Let $S \in \mathbb{F}_2^n$.*

- 1) *If $wt(S)$ is even, then $W_{F_n}(S) = 0$.*
- 2) *If $wt(S)$ is odd, then*

$$W_{F_n}(S) = (-1)^{(wt(S)+1)/2} 2^{\binom{n-1}{t}} \prod_{i=1}^{(wt(S)-1)/2} \frac{2i-1}{n-2i}.$$

Proof. Since $\sum_{wt(X)=i} (-1)^{S \cdot X} = K_i(wt(S), n)$, we have

$$W_{F_n}(S) = \sum_{i=t+1}^n K_i(wt(S), n) - \sum_{i=0}^t K_i(wt(S), n), \tag{6}$$

where $K_i(k, n)$ is the so-called Krawtchouk polynomial [15, Page 151, Part I] defined by

$$K_i(k, n) = \sum_{j=0}^i (-1)^j \binom{k}{j} \binom{n-k}{i-j}, i = 0, 1, \dots, n.$$

Krawtchouk polynomials also have properties [15, Page 153, Part I] as follows.

P1. $K_i(k, n) = (-1)^k K_{n-i}(k, n)$.

P2. $\sum_{i=0}^e K_i(k, n) = K_e(k-1, n-1)$.

P3. $(n-k)K_i(k+1, n) = (n-2i)K_i(k, n) - kK_i(k-1, n)$ for nonnegative integers i and k .

If $wt(S)$ is even, then by (6) and P1, we have $W_{F_n}(S) = 0$.

If $wt(S)$ is odd, then by (6), P1 and P2, we have

$$W_{F_n}(S) = -2 \sum_{i=0}^t K_i(wt(S), n) = -2K_t(wt(S) - 1, n - 1).$$

By the definition of Krawtchouk polynomials, we have $K_t(k, n - 1) = 0$ if k is odd. Thus by P3, we have

$$\begin{aligned}
W_{F_n}(S) &= (-1)^{(wt(S)-1)/2+1} 2K_t(0, n-1) \prod_{i=1}^{(wt(S)-1)/2} \frac{2i-1}{n-2i} \\
&= (-1)^{(wt(S)+1)/2} 2 \binom{n-1}{t} \prod_{i=1}^{(wt(S)-1)/2} \frac{2i-1}{n-2i}.
\end{aligned}$$

Lemma 3. Let $S, T \in \mathbb{F}_2^n$.

1) If $wt(S) + wt(T) = n + 1$, then $W_{F_n}(S) = (-1)^t W_{F_n}(T)$.

2) If both $wt(S)$ and $wt(T)$ are odd, and $0 < wt(S) < wt(T) \leq t + 1$, then $|W_{F_n}(S)| > |W_{F_n}(T)|$.

Proof. 1) Since Krawtchouk polynomials have the following property,

$$K_i(k, n) = (-1)^i K_i(n - k, n),$$

we have that

$$\begin{aligned}
W_{F_n}(S) &= -2K_t(wt(S) - 1, n - 1) \\
&= -2(-1)^t K_t(n - 1 - (wt(S) - 1), n - 1) \\
&= -2(-1)^t K_t(wt(T) - 1, n - 1) = (-1)^t W_{F_n}(T).
\end{aligned}$$

2) It is obvious from the second item of Lemma 2.

Remark 2. By Lemma 3, we have

$$\max_{T \in \mathbb{F}_2^n} |W_{F_n}(T)| = |W_{F_n}(S_1)| = |W_{F_n}(S_n)| = 2 \binom{n-1}{t},$$

where $wt(S_1) = 1$, $wt(S_n) = n$. Therefore, $nl(F_n) = 2^{n-1} - \binom{n-1}{t}$ [9]. And

$$\max_{T \in \mathbb{F}_2^n, wt(T) \neq 1, n} |W_{F_n}(T)| = |W_{F_n}(S_3)| = |W_{F_n}(S_{n-2})| = \frac{2}{n-2} \binom{n-1}{t},$$

where $wt(S_3) = 3$, $wt(S_{n-2}) = n - 2$. We note that the difference between the maximal and the secondarily maximal absolute value of Walsh spectra is quite great, which is

$$2 \frac{n-3}{n-2} \binom{n-1}{t}.$$

Algebraic immunity has the following relationship with nonlinearity.

Lemma 4. [14] Let f be an n -variable Boolean function, $AI(f) = k$, then

$$nl(f) \geq 2^{n-1} - \sum_{i=k-1}^{n-k} \binom{n-1}{i},$$

and this bound is tight.

Remark 3. Lemma 4 together with Remark 2 implies that F_n has the worst nonlinearity among all n -variable Boolean functions with maximum AI.

Theorem 6. *The Walsh spectra of $f = f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is given by*

$$W_f(S) = W_{F_n}(S) - 4\left(\sum_{r=1}^k S \cdot X_{j_r} - \sum_{r=1}^k S \cdot Y_{i_r}\right).$$

Proof

$$\begin{aligned} W_f(S) &= \sum_{r=1}^{2^n-1} (-1)^{f(X_r)+S \cdot X_r} + \sum_{r=1}^{2^n-1} (-1)^{f(Y_r)+S \cdot Y_r} \\ &= \sum_{r \in \{1, \dots, 2^n-1\} \setminus \{j_1, \dots, j_k\}} (-1)^{F_n(X_r)+S \cdot X_r} + \sum_{r=1}^k (-1)^{F_n(X_{j_r})+1+S \cdot X_{j_r}} + \\ &\quad \sum_{r \in \{1, \dots, 2^n-1\} \setminus \{i_1, \dots, i_k\}} (-1)^{F_n(Y_r)+S \cdot Y_r} + \sum_{r=1}^k (-1)^{F_n(Y_{i_r})+1+S \cdot Y_{i_r}} \\ &= W_{F_n}(S) - 2\left(\sum_{r=1}^k (-1)^{F_n(X_{j_r})+S \cdot X_{j_r}} + \sum_{r=1}^k (-1)^{F_n(Y_{i_r})+S \cdot Y_{i_r}}\right) \\ &= W_{F_n}(S) - 2\left(\sum_{r=1}^k (-1)^{1+S \cdot X_{j_r}} + \sum_{r=1}^k (-1)^{S \cdot Y_{i_r}}\right) \\ &= W_{F_n}(S) - 2\left(\sum_{r=1}^k (2S \cdot X_{j_r} - 1) + \sum_{r=1}^k (1 - 2S \cdot Y_{i_r})\right) \\ &= W_{F_n}(S) - 4\left(\sum_{r=1}^k S \cdot X_{j_r} - \sum_{r=1}^k S \cdot Y_{i_r}\right). \end{aligned}$$

From the above analysis in this section, some necessary conditions of Boolean functions with maximum AI and these functions which also have higher nonlinearity than that of F_n can be obtained.

Theorem 7. *Let $1 \leq k \leq 2^{n-1}$, $1 \leq i_1 < \dots < i_k \leq 2^{n-1}$, $1 \leq j_1 < \dots < j_k \leq 2^{n-1}$. If one of the following conditions is satisfied, then $AI(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) < t + 1$.*

- 1) *There exists $1 \leq r \leq n$, such that $x_{j_1, r} + \dots + x_{j_k, r} > y_{i_1, r} + \dots + y_{i_k, r}$.*
- 2) *If $n \equiv 1 \pmod 4$,*

$$\#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) \text{ is odd}\} > \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is odd}\};$$

if $n \equiv 3 \pmod 4$,

$$\#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) \text{ is odd}\} < \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is odd}\}.$$

Proof. By Theorem 6, the first condition means that $|W_{f_{(i_1, \dots, i_k; j_1, \dots, j_k)}}(S)| > |W_{F_n}(S)|$ for $S = (\underbrace{0, \dots, 0}_{r-1}, 1, 0, \dots, 0)$. Thus, we have $nl(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) < nl(F_n)$ by Remark 2. Therefore, by Remark 3, we have $AI(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) < t + 1$.

If the second condition is satisfied, then $|W_{f_{(i_1, \dots, i_k; j_1, \dots, j_k)}}(S)| > |W_{F_n}(S)|$ for $S = (1, 1, \dots, 1)$. In the same way, the result can be proved.

Theorem 8. *Let $f = f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ be an n -variable Boolean function with $AI \ t + 1$. If one of the following conditions is satisfied, then f has the worst nonlinearity among all n -variable Boolean functions with maximum AI .*

- 1) *There exists $1 \leq r \leq n$, such that $x_{j_1, r} + \dots + x_{j_k, r} = y_{i_1, r} + \dots + y_{i_k, r}$.*
- 2) $\#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) \text{ is odd}\} = \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is odd}\}$.

Proof. By Theorem 6, the first condition means that $|W_{f_{(i_1, \dots, i_k; j_1, \dots, j_k)}}(S)| = |W_{F_n}(S)|$ for $S = (\underbrace{0, \dots, 0}_{r-1}, 1, 0, \dots, 0)$. Thus, we have $nl(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) \leq nl(F_n)$ by Remark 2. Therefore, by Remark 3, we have $nl(f_{(i_1, \dots, i_k; j_1, \dots, j_k)}) = nl(F_n)$, and the result is proved.

If the second condition is satisfied, then $|W_{f_{(i_1, \dots, i_k; j_1, \dots, j_k)}}(S)| = |W_{F_n}(S)|$ for $S = (1, 1, \dots, 1)$. In the same way, the result can be proved.

Corollary 3. *For any $1 \leq i, j \leq 2^{n-1}$, if $AI(f_{(i;j)}) = t + 1$ then $f_{(i;j)}$ has the worst nonlinearity among all n -variable Boolean functions with maximum AI .*

Proof. From Theorem 8, it is sufficient to consider the case of $i = 2^{n-1}, j = 1$, i.e. $X = (0, 0, \dots, 0), Y = (1, 1, \dots, 1)$. In this case, from the first item of Corollary 1 we have $AI(f_{(i;j)}) < t + 1$ which contradicts the assumption.

Theorem 9. *If $1 \leq k \leq \frac{n-3}{4(n-2)} \binom{n-1}{t}$, then $nl(f_{(i_1, \dots, i_k; j_1, \dots, j_k)})$ is given by*

$$2^{n-1} - \binom{n-1}{t} + 2 \min \left\{ \min_{1 \leq s \leq n} \left(\sum_{r=1}^k y_{i_r, s} - \sum_{r=1}^k x_{j_r, s} \right), (-1)^t (N_1 - N_2) \right\},$$

where

$$N_1 = \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is odd}\},$$

$$N_2 = \#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) \text{ is odd}\}.$$

Proof. Denote $f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ by f . From Theorem 6 we have,

$$|W_{F_n}(S)| - 4k \leq |W_f(S)| \leq |W_{F_n}(S)| + 4k.$$

Let $S, T \in \mathbb{F}_2^n$, and $wt(S) = 1$ or n , $wt(T) \notin \{1, n\}$. If $1 \leq k \leq \frac{n-3}{4(n-2)} \binom{n-1}{t}$, then by Remark 2,

$$|W_f(S)| \geq |W_{F_n}(S)| - 4k \geq |W_{F_n}(T)| + 4k \geq |W_f(T)|.$$

Therefore, we have $\max_{T \in \mathbb{F}_2^n} |W_f(T)| = \max_{wt(S)=1, n} |W_f(S)|$.

Case 1. $wt(S) = 1$ and $S = (\underbrace{0, \dots, 0}_{s-1}, 1, 0, \dots, 0)$. By Theorem 6 we have

$$|W_f(S)| = 2 \binom{n-1}{t} - 4 \left(\sum_{r=1}^k y_{i_r, s} - \sum_{r=1}^k x_{j_r, s} \right).$$

Case 2. $wt(S) = n$. By Theorem 6 we have

$$|W_f(S)| = 2 \binom{n-1}{t} - 4((-1)^t(N_1 - N_2)).$$

Hence the result holds from $nl(f) = 2^{n-1} - \frac{1}{2} \max_{S \in \mathbb{F}_2^n} |W_f(S)|$.

Now, we modify Construction 1 to construct n -variable Boolean functions with maximum AI and possibly having higher nonlinearity.

Construction 2. Step1: Select randomly an integer $1 \leq k \leq 2^{n-2}$ and k integers $1 \leq i_1 < \dots < i_k \leq 2^{n-1}$, which satisfy

- i) $\min_{1 \leq s \leq n} \sum_{r=1}^k y_{i_r, s}$ is as large as possible;
- ii) if $n \equiv 1 \pmod 4$, $\#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is odd} \}$ is as large as possible; if $n \equiv 3 \pmod 4$, $\#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is even} \}$ is as large as possible.

Step2: Find out k integers $1 \leq j_1 < \dots < j_k \leq 2^{n-1}$, which satisfies

- i) the j_1 th, \dots , j_k th column vectors of $W_{(i_1, \dots, i_k)}$ are linearly independent;
- ii) $a = \min_{1 \leq s \leq n} \left(\sum_{r=1}^k y_{i_r, s} - \sum_{r=1}^k x_{j_r, s} \right)$ is as large as possible;
- iii) if $n \equiv 1 \pmod 4$,

$$b = \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is odd} \} - \#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) \text{ is odd} \}$$

is as large as possible; if $n \equiv 3 \pmod 4$,

$$c = \#\{X \in \{X_{j_1}, \dots, X_{j_k}\} | wt(X) \text{ is odd} \} - \#\{Y \in \{Y_{i_1}, \dots, Y_{i_k}\} | wt(Y) \text{ is odd} \}$$

is as large as possible.

Then, the Boolean function $f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ defined by (1) has AI $t + 1$ and has possibly a higher nonlinearity.

Remark 4. From Theorem 9, the function obtained by Construction 2 will has a higher nonlinearity than that of F_n if $1 \leq k \leq \frac{n-3}{4(n-2)} \binom{n-1}{t}$ and $a > 0$, $b > 0$ (if $n \equiv 1 \pmod 4$) or $c > 0$ (if $n \equiv 3 \pmod 4$), and it possibly has a nonlinearity equal to that of F_n if $k > \frac{n-3}{4(n-2)}$.

Further, the following theorem provides a class of n -variable Boolean functions with maximum AI which also have higher nonlinearity than that of F_n .

Theorem 10. Let $n \equiv 3 \pmod{4}$, $1 \leq k \leq \min\{n, \frac{n-3}{4(n-2)} \binom{n-1}{t}\}$, Y_{i_1}, \dots, Y_{i_k} belong to 0_{F_n} and their Hamming weights are w_1, \dots, w_k , respectively. If

$$1) \bigoplus_{r=0}^{t-1} \binom{w_i-1}{r} = 1, \quad i = 1, \dots, k; \text{ and}$$

2) w_1, \dots, w_k are not all odd; and

3) there exist $1 \leq j_1 < \dots < j_k \leq n$, such that the j_1 th, \dots , j_k th columns of

the matrix $\begin{pmatrix} Y_{i_1} \\ \dots \\ Y_{i_k} \end{pmatrix}$ are linearly independent; and

4) for any $s \notin \{j_1, \dots, j_k\}$, $y_{i_1,s} + \dots + y_{i_k,s} \geq 1$; and for any $s \in \{j_1, \dots, j_k\}$, $y_{i_1,s} + \dots + y_{i_k,s} \geq 2$.

then, $AI(f_{(i_1, \dots, i_k; j_1+1, \dots, j_k+1)}) = t+1$ and $nl(f_{(i_1, \dots, i_k; j_1+1, \dots, j_k+1)}) \geq nl(F_n) + 2$.

Example 5. The Boolean function defined in Example 4 has AI 4. And $nl(f) = nl(F_n) + 2$.

Finally, we obtain the following sufficient and necessary condition of Boolean functions with maximum AI which are also resilient functions.

Theorem 11. Let $f = f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ be an n -variable Boolean function. Then, f is 1-resilient function if and only if

$$\sum_{r=1}^k y_{i_r, s} - \sum_{r=1}^k x_{j_r, s} = \frac{1}{2} \binom{n-1}{t},$$

for $s = 1, \dots, n$.

Corollary 4. Let $f = f_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ be an n -variable Boolean function. Then, f is 1-resilient function and has AI $t+1$ if and only if

$$\sum_{r=1}^k y_{i_r, s} - \sum_{r=1}^k x_{j_r, s} = \frac{1}{2} \binom{n-1}{t},$$

for $s = 1, \dots, n$, and $W_{(i_1, \dots, i_k; j_1, \dots, j_k)}$ is invertible.

6 Conclusion

Possessing a high algebraic immunity is a necessary condition for Boolean functions used in stream ciphers against algebraic attacks. In this paper, some classes of $(2t+1)$ -variable Boolean functions with maximum AI are obtained. Further, some necessary conditions of this kind of functions which also have higher non-linearity are presented and thus a modified construction method is proposed to obtain such functions. Finally, a sufficient and necessary condition of $(2t+1)$ -variable Boolean functions with maximum AI which are also 1-resilient is presented. However, it is still open that what is the highest nonlinearity of Boolean functions with maximum AI and how to construct Boolean functions which have maximum AI and the highest nonlinearity.

References

1. F. Armknecht. Improving fast algebraic attacks. In *FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 65-82. Springer-Verlag, 2004.
2. F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 162-175. Springer-Verlag, 2003.
3. A. Braeken and B. Preneel. On the algebraic immunity of symmetric Boolean functions. In *INDOCRYPT 2005*, volume 3797 of *Lecture Notes in Computer Science*, pages 35-48. Springer-Verlag, 2005.
4. C. Carlet. A method of construction of balanced functions with optimum algebraic immunity. Available at <http://eprint.iacr.org/2006/149>.
5. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176-194. Springer-Verlag, 2003.
6. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345-359. Springer-Verlag, 2003.
7. D. K. Dalai, K. C. Gupta and S. Maitra. Results on algebraic immunity for cryptographically significant Boolean functions. In *INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 92-106. Springer-Verlag, 2004.
8. D. K. Dalai, K. C. Gupta and S. Maitra. Cryptographically significant Boolean functions: construction and analysis in terms of algebraic immunity. In *FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 98-111. Springer-Verlag, 2005.
9. D. K. Dalai, S. Maitra and S. Sarkar. Basic theory in construction of Boolean functions with maximum possible annihilator immunity. *Designs, Codes and Cryptography*, 40:41-58, 2006.
10. D. K. Dalai and S. Maitra. Reducing the Number of Homogeneous Linear Equations in Finding Annihilators. Available at <http://eprint.iacr.org/2006/032>.
11. C. Ding, G. Xiao and W. Shan. *The stability theory of stream ciphers*. Springer-Verlag, 1991.
12. N. Li and W. F. Qi. Symmetric Boolean functions depending on an odd number of variables with maximum algebraic immunity. *IEEE Transaction on Information Theory*, 52(5):2271-2273, May 2006.
13. N. Li and W. F. Qi. Construction and count of Boolean functions of an odd number of variables with maximum algebraic immunity. Available at <http://arxiv.org/abs/cs.CR/0605139>.
14. M. Lobanov. Tight bound between nonlinearity and algebraic immunity. Available at <http://eprint.iacr.org/2005/441>.
15. F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*. Elsevier, North-Holland, 1977.
16. W. Meier, E. Pasalic and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474-491. Springer-Verlag, 2004.

Secure Sketch for Biometric Templates

Qiming Li¹, Yagiz Sutcu², and Nasir Memon³

¹ Department of Computer and Information Science

² Department of Electrical and Computer Engineering

³ Department of Computer and Information Science

Polytechnic University

6 Metrotech Center, Brooklyn, NY 11201

qiming.li@ieee.org, ygzstc@yahoo.com, memon@poly.edu

Abstract. There have been active discussions on how to derive a consistent cryptographic key from noisy data such as biometric templates, with the help of some extra information called a *sketch*. It is desirable that the sketch reveals little information about the biometric templates even in the worst case (i.e., the *entropy loss* should be low). The main difficulty is that many biometric templates are represented as points in continuous domains with unknown distributions, whereas known results either work only in discrete domains, or lack rigorous analysis on the entropy loss. A general approach to handle points in continuous domains is to quantize (discretize) the points and apply a known sketch scheme in the discrete domain. However, it can be difficult to analyze the entropy loss due to quantization and to find the “optimal” quantizer. In this paper, instead of trying to solve these problems directly, we propose to examine the *relative entropy loss* of any given scheme, which bounds the number of additional bits we could have extracted if we used the optimal parameters. We give a general scheme and show that the relative entropy loss due to suboptimal discretization is at most $(n \log 3)$, where n is the number of points, and the bound is tight. We further illustrate how our scheme can be applied to real biometric data by giving a concrete scheme for face biometrics.

Keywords: Secure sketch, biometric template, continuous domain.

1 Introduction

The main challenge in using biometric data in cryptography is that they cannot be reproduced exactly. Some noise will be inevitably introduced into biometric samples during acquisition and processing. There have been active discussions on how to extract a reliable cryptographic key from such noisy data. Some recent techniques attempt to correct the noise in the data by using some public information P derived from the original biometric template X . These techniques include fuzzy commitment [12], fuzzy vault [11], helper data [19], and secure sketch [7]. In this paper, we follow Dodis et al. [7] and call such public information P a *sketch*.

Typically, there are two main components in a secure sketch scheme. The first is the sketch generation algorithm, which we will refer to as the *encoder*. It takes the original biometric template X as the input, and outputs a sketch P . The second algorithm is the biometric template reconstruction algorithm, or the *decoder*, which takes another biometric template Y and the sketch P as the input and outputs X' . If Y and X are sufficiently similar according to some similarity measure, we will have $X = X'$. An important requirement for such a scheme is that the sketch P should not reveal too much information about the biometric template X . Dodis et al. [7] gives a notion of *entropy loss*, which (informally speaking) measures the advantage that P gives to any adversary in guessing X , when X is discrete in nature (Section 3 provides the details). It is worth to note that the entropy loss is a worst case bound for *all* distributions of X .

There are several difficulties in applying many known secure sketch techniques to known types of biometric templates directly. Firstly, many biometric templates are represented by sequences of n points in a continuous domain (say, \mathbb{R}), or equivalently, points in an n -dimensional space (say, \mathbb{R}^n). In this case, since the entropy of the original data can be very large, and the length of the extracted key is typically quite limited, the “entropy loss” as defined in [7] can be very high for any possible scheme. For example, X is often a discrete approximation of some points in a continuous domain (e.g., decimal fractions obtained by rounding real numbers). As the precision of X gets higher, both the entropy of X and the entropy loss from P become larger, but the extracted key can become stronger. Hence, this notion of entropy loss alone is insufficient, and the seemingly high entropy loss for this type of biometric data would be misleading. We will discuss this issue in detail in Section 4, and give a complimentary definition of *relative entropy loss* for noisy data in the continuous domain. Informally speaking, the relative entropy loss of a sketch measures the imperfectness of the rounding, which is the maximum amount of additional entropy we can obtain by the “optimal” rounding. At the same time, the entropy loss from P serves as a measure of the security of the sketch in the discrete domain.

Secondly, even if the biometric templates are represented in discrete form, there are practical problems when the entropy of the original template is high. For example, the iris pattern of an eye can be represented by a 2048 bit binary string called *iris code*, and up to 20% of the bits could be changed under noise [9]. The fuzzy commitment scheme based on binary error-correcting codes [12] seems to be applicable at the first glance. However, it would be impractical to apply a binary error-correcting code on such a long string with such a large error-correcting capability. A two-level error-correcting technique is proposed in [9], which essentially changes the similarity measure. As a result, the space is no longer a metric space.

Thirdly, the similarity measures for many known biometric templates can be quite different from those considered in many theoretical works (such as Hamming distance, set difference and edit distance in [7]). This can happen as a result of technical considerations (e.g., in the case of iris codes). However, in many cases this is due to the nature of biometric templates. For instance,

a fingerprint template usually consists of a set of minutiae (feature points in 2-D space), and two templates are considered as similar if more than a certain number of minutiae in one template are near distinct minutiae in the other. In this case, the similarity measure has to consider both Euclidean distance and set difference at the same time.

The secure sketch for point sets [5] is perhaps the first rigorous approach to similarity measures that do not define a metric space. A generic scheme is proposed in [5] for point sets in bounded discrete d -dimensional space for any d , where the underlying similarity measure is motivated by the similarity measure of fingerprint templates. While such a scheme is potentially applicable to fingerprints represented as minutiae, other types of biometrics are different both in representations and similarity measures, thus require different considerations and different schemes.

In this paper, we study how to design secure sketch for biometric templates, where the worst case bound can be proved. We observe that many biometric templates can be represented in a general form: The original X can be considered as a list of n points, where each point x of X is in a bounded continuous domain. Under noise, each point can be perturbed by a distance less than δ , and on top of that, at most t points can be replaced. Similar to [5], we will refer to the first noise as the *white noise*, and the second *replacement noise*. We note that this similarity measure can be applied to handwritten online signatures [8], iris patterns [9], voice features [15], and face biometrics [17]. This formulation is different from that in [5] in two ways: (1) The points are in a continuous domain, and (2) the points are always ordered.

To handle points in continuous domain, a general two step approach is to (1) quantize (i.e., discretize) the points in X to a discrete domain with a scalar quantizer \mathcal{Q}_λ , where λ is the step size, and (2) apply secure sketch techniques on the quantized points $\hat{X} = \mathcal{Q}_\lambda(X)$ in the quantized domain, which is discrete. For example, if points in X are real numbers between 0 and 1, assume that we have a scalar quantizer \mathcal{Q}_λ with step size $\lambda = 0.01$, such that $\mathcal{Q}_\lambda(x) = \hat{x}$ if and only if $\hat{x}\lambda \leq x < (\hat{x} + 1)\lambda$, then every point in X would be mapped to an integer in $[0, 99]$. After that, we can apply a secure sketch for discrete points in the domain $[0, 99]^n$ to achieve error-tolerance.

However, there are two difficulties when this approach is applied. Firstly, if we follow the notion of secure sketch and entropy loss as in [7], the quantization error $X - \hat{X}$ in the first step has to be kept in the sketch, since exact reconstruction of X is required by definition. However, it can be difficult to give an upper bound on the entropy loss from the quantization errors. Even if we can, it can be very large.

Furthermore, as the quantization step λ becomes very small, the bound on the entropy loss in the quantized domain during the second step can be very high. For instance, for $x \in [0, 1)$ and $\delta = 0.01$, when $\lambda = 0.01$, the entropy loss in Step (2) will be $\log 3$, and the bound is tight. When $\lambda = 0.001$, the entropy loss will be $\log 21$. However, the big difference in entropy loss in the quantized domain can be misleading. We will revisit this example in Section 5, and will show that the second case actually results in a stronger key if X is uniformly distributed.

To address the above problems, we consider the following strategy. Instead of trying to answer the question of how much entropy is lost during quantization, we study how different quantizers affect the strength of the key that we can finally extract from the noisy data. In particular, given a secure sketch scheme in the discrete domain and a quantizer \mathcal{Q}_1 with step size λ_1 , we consider any quantizer \mathcal{Q}_2 with step size λ_2 . Assuming that m_1 and m_2 are the strengths of the keys under these two quantizers respectively, we found that it is possible to give an upper bound on the difference between m_1 and m_2 , for any distribution of X , and any choices of λ_2 (hence \mathcal{Q}_2) within a certain range. This bound can be expressed as a function of λ_1 . In other words, although we do not know what is the exact entropy loss due to the quantizer \mathcal{Q}_1 , we do know that at most how far away \mathcal{Q}_1 can be from the “optimal” one. Based on this, we give a notion of *relative entropy loss* for data in continuous domain. Furthermore, we show that if X is uniformly distributed, the relative entropy loss can be bounded by a constant for any choice of λ_1 .

To illustrate how our general approach can be applied to practical biometric templates, we give a scheme based on the authentication scheme for face biometrics in [17]. We will also discuss some practical issues in designing secure sketch schemes for biometric templates.

We note that our proposed schemes and analysis can be applied for two parties to extract secret keys given correlated random variables (e.g., [14]), where the random variables take values in a continuous domain (e.g. \mathbb{R}). The entropy loss in the quantized domain measures how much information can be leaked to an eavesdropper, while the relative entropy loss measures how many additional bits that we might be able to extract.

We will give a review of related works in Section 2, followed by some preliminary formal definitions in Section 3. Our definition of secure sketch and its security will be presented in Section 4. We give a general similarity measure and our proposed schemes in Section 5, together with a security analysis and some discussions on choosing the parameters. A concrete secure sketch scheme for face biometrics will be given in 6.

2 Related Works

It is not surprising that the construction of the sketch largely depends on the representation of the biometric templates and the underlying distance function that measures the similarity. Most of the known techniques assume that the noisy data under consideration are represented as points in some metric space. The fuzzy commitment scheme [12], which is based on binary error-correcting codes, considers binary strings where the similarity is measured by Hamming distance. The fuzzy vault scheme [11] considers sets of elements in a finite field with set difference as the distance function, and corrects errors by polynomial interpolation. Dodis et al. [7] further gives the notion of *fuzzy extractors*, where a “strong extractor” (such as pair-wise independent hash functions) is applied after the original X is reconstructed to obtain an almost uniform key. Constructions

and rigorous analysis of secure sketch are given in [7] for three metrics: Hamming distance, set difference and edit distance. Secure sketch schemes for point sets in [5] are motivated by the typical similarity measure used for fingerprints, where each template consists of a set of points in 2-D space, and the similarity measure does not define a metric space.

On the other hand, there have been a number of works on how to extract consistent keys from real biometric templates, which have quite different representations and similarity measures from the above theoretical works. Such biometric templates include handwritten online signatures [8], fingerprints [20], iris patterns [9], voice features [15], and face biometrics [17]. These works, however, do not have sufficiently rigorous treatment of the security, compared to well-established cryptographic techniques. Some of the works give analysis on the entropy of the biometrics, and approximated amount of efforts required by a brute-force attacker.

Boyen [2] shows that a sketch scheme that is provably secure may be insecure when multiple sketches of the same biometric data are obtained. Boyen et al. further study the security of secure sketch schemes under more general attacker models in [1], and techniques to achieve mutual authentication are proposed.

Linnartz and Tuyls [13] consider a similar problem for biometric authentication applications. They consider zero mean i.i.d. jointly Gaussian random vectors as biometric templates, and use mutual information as the measure of security against dishonest verifiers. Tuyls and Goseling [19] consider a similar notion of security, and develop some general results when the distribution of the original is known and the verifier can be trusted. Some practical results along this line also appear in [18].

3 Preliminaries

3.1 Entropy and Entropy Loss in Discrete Domain

In the case where X is discrete, we follow the definitions by Dodis et al. [7]. They consider a variant of the *average min-entropy* of X given P , which is essentially the minimum strength of the key that can be consistently extracted from X when P is made public.

In particular, the min-entropy $\mathbf{H}_\infty(A)$ of a discrete random variable A is defined as $\mathbf{H}_\infty(A) = -\log(\max_a \Pr[A = a])$. For two discrete random variables A and B , the average min-entropy of A given B is defined as $\tilde{\mathbf{H}}_\infty(A | B) = -\log(\mathbb{E}_{b \leftarrow B}[2^{-\mathbf{H}_\infty(A|B=b)}])$.

For discrete X , the entropy loss of the sketch P is defined as $\mathcal{L} = \mathbf{H}_\infty(X) - \tilde{\mathbf{H}}_\infty(X|P)$. This definition is useful in the analysis, since for any ℓ -bit string B , we have $\tilde{\mathbf{H}}_\infty(A | B) \geq \mathbf{H}_\infty(A) - \ell$. For any secure sketch scheme for discrete X , let R be the randomness invested in constructing the sketch, it is not difficult to show that when R can be computed from X and P , we have

$$\mathcal{L} = \mathbf{H}_\infty(X) - \tilde{\mathbf{H}}_\infty(X | P) \leq |P| - \mathbf{H}_\infty(R). \quad (1)$$

In other words, the entropy loss can be bounded from above by the difference between the size of P and the amount of randomness we invested in computing P . This allows us to conveniently find an upper bound of \mathcal{L} for any distribution of X , since it is independent of X .

3.2 Secure Sketch in Discrete Domain

Our definitions of secure sketch and entropy loss in the discrete domain follow that in [7]. Let \mathcal{M} be a finite set of points with a *similarity* relation $\mathbf{S} \subseteq \mathcal{M} \times \mathcal{M}$. When $(X, Y) \in \mathbf{S}$, we say the Y is similar to X , or the pair (X, Y) is similar.

Definition 1. *A sketch scheme in discrete domain is a tuple $(\mathcal{M}, \mathbf{S}, \text{Enc}, \text{Dec})$, where $\text{Enc} : \mathcal{M} \rightarrow \{0, 1\}^*$ is an encoder and $\text{Dec} : \mathcal{M} \times \{0, 1\}^* \rightarrow \mathcal{M}$ is a decoder such that for all $X, Y \in \mathcal{M}$, $\text{Dec}(Y, \text{Enc}(X)) = X$ if $(X, Y) \in \mathbf{S}$. The string $P = \text{Enc}(X)$ is the sketch, and is to be made public. We say that the scheme is \mathcal{L} -secure if for all random variables X over \mathcal{M} , the entropy loss of the sketch P is at most \mathcal{L} . That is, $\mathbf{H}_\infty(X) - \tilde{\mathbf{H}}_\infty(X | \text{Enc}(X)) \leq \mathcal{L}$.*

We call $\tilde{\mathbf{H}}_\infty(X | P)$ the *left-over entropy*, which in essence measures the “strength” of the key that can be extracted from X given that P is made public. Note that in most cases, the ultimate goal is to maximize the left-over entropy for some particular distribution of X . However, in the discrete case, the min-entropy of X is fixed but can be difficult to analyze. Hence, entropy loss becomes an equivalent measure which is easier to quantify.

4 Secure Sketch in Continuous Domain

In this section we propose a general approach to handle noisy data in a continuous domain. We consider points in a universe \mathcal{U} , which is a set that may be uncountable. Let \mathbf{S} be a similarity relation on \mathcal{U} , i.e., $\mathbf{S} \subseteq \mathcal{U} \times \mathcal{U}$. Let \mathcal{M} be a set of finite points, and let $\mathcal{Q} : \mathcal{U} \rightarrow \mathcal{M}$ be a function that maps points in \mathcal{U} to points in \mathcal{M} . We will refer to such a function \mathcal{Q} as a *quantizer*.

Definition 2. *A quantization-based sketch scheme is a tuple $(\mathcal{U}, \mathbf{S}, \mathcal{Q}, \mathcal{M}, \text{Enc}, \text{Dec})$, where $\text{Enc} : \mathcal{M} \rightarrow \{0, 1\}^*$ is an encoder and $\text{Dec} : \mathcal{M} \times \{0, 1\}^* \rightarrow \mathcal{M}$ is a decoder such that for all $X, Y \in \mathcal{U}$, $\text{Dec}(\mathcal{Q}(Y), \text{Enc}(\mathcal{Q}(X))) = \mathcal{Q}(X)$ if $(X, Y) \in \mathbf{S}$. The string $P = \text{Enc}(\mathcal{Q}(X))$ is the sketch. We say that the scheme is \mathcal{L} -secure in the quantized domain if for all random variable X over \mathcal{U} , the entropy loss of P is at most \mathcal{L} , i.e., $\mathbf{H}_\infty(\mathcal{Q}(X)) - \tilde{\mathbf{H}}_\infty(\mathcal{Q}(X) | \text{Enc}(\mathcal{Q}(X))) \leq \mathcal{L}$.*

In other words, a quantization is applied to transform the points in the continuous domain to a discrete domain, and a sketch scheme for discrete domain is applied to obtain the sketch P . During reconstruction, we require the exact reconstruction of the quantization $\mathcal{Q}(X)$ instead of the original X in the continuous domain. When required, a strong extractor can be further applied to $\mathcal{Q}(X)$ to extract a key (as the fuzzy extractor in [7]). That is, we treat $\mathcal{Q}(X)$ as the “discrete original”. Similarly, we call $\tilde{\mathbf{H}}_\infty(\mathcal{Q}(X) | P)$ the left-over entropy.

When \mathcal{Q} is fixed, we can use the entropy loss on $\mathcal{Q}(X)$ to analyze the security of the scheme, and bound the entropy loss of P . However, using this entropy loss alone may be misleading, since there are many ways to quantize X , and different quantizer would make a difference in both the min-entropy of $\mathcal{Q}(X)$ and the entropy loss. Since our ultimate goal is to maximize the left-over entropy (i.e., the average min-entropy $\tilde{\mathbf{H}}_\infty(\mathcal{Q}(X) | P)$), the entropy loss alone is not sufficient to compare different quantization strategies.

To illustrate the subtleties, we consider the following example. Let x be a point uniformly distributed in the interval $[0, 1]$, and under noise, it can be shifted but still within the range $[x - 0.01, x + 0.01]$. We can use a scalar quantizer \mathcal{Q}_1 with step size 0.01, such that all points in the interval $[0, 1]$ are mapped to integers $[0, 99]$. In this case, the min-entropy $\mathbf{H}_\infty(\mathcal{Q}_1(x)) = \log 100$. As we can see later, there is an easy way to construct a secure sketch for such $\mathcal{Q}_1(x)$ with entropy loss of $\log 3$. Hence, the left-over entropy is $\log(100/3) \approx 5.06$. Now we consider another scalar quantizer \mathcal{Q}_2 with step size 0.001, such that the range of $\mathcal{Q}_2(x)$ is $[0, 999]$. A similar scheme on $\mathcal{Q}_2(x)$ would give entropy loss of $\log 21$, which seems much larger than the previous $\log 3$. However, the min-entropy of $\mathcal{Q}_2(x)$ is also increased to $\log 1000$, and the left-over entropy would be $\log(1000/21) \approx 5.57$, which is slightly higher than the case where \mathcal{Q}_1 is used.

Intuitively, for a given class of methods of handling noisy data in the quantized domain, it is important to examine how different precisions of the quantization process affect the strength of the extracted key. For this purpose, we propose to consider not just one, but a family of quantizers \mathbf{Q} , where each quantizer \mathcal{Q} drawn from \mathbf{Q} defines a mapping from \mathcal{U} to a finite set $\mathcal{M}_\mathcal{Q}$. Let \mathbf{M} be the set of such $\mathcal{M}_\mathcal{Q}$ for all $\mathcal{Q} \in \mathbf{Q}$. We also define a family of encoders \mathbf{E} and decoders \mathbf{D} , such that for each \mathcal{Q} and $\mathcal{M}_\mathcal{Q}$, there exist uniquely defined $\text{Enc}_\mathcal{Q} \in \mathbf{E}$ and $\text{Dec}_\mathcal{Q} \in \mathbf{D}$ that can handle $\mathcal{Q}(X)$ in $\mathcal{M}_\mathcal{Q}$.

Definition 3. *A quantization-based sketch family is a tuple $(\mathcal{U}, \mathbf{S}, \mathbf{Q}, \mathbf{M}, \mathbf{E}, \mathbf{D})$, such that for each quantizer $\mathcal{Q} \in \mathbf{Q}$, there exist $\mathcal{M} \in \mathbf{M}$, $\text{Enc} \in \mathbf{E}$ and $\text{Dec} \in \mathbf{D}$, and $(\mathcal{U}, \mathbf{S}, \mathcal{Q}, \mathcal{M}, \text{Enc}, \text{Dec})$ is a quantization-based sketch scheme. We say that such a scheme is a member of the family, and is identified by \mathcal{Q} .*

Definition 4. *A quantization-based sketch family $(\mathcal{U}, \mathbf{S}, \mathbf{Q}, \mathbf{M}, \mathbf{E}, \mathbf{D})$ is (\mathbf{L}, \mathbf{R}) -secure for functions $\mathbf{L}, \mathbf{R} : \mathbf{Q} \rightarrow \mathbb{R}$ if for any member identified by \mathcal{Q}_1 (with encoder Enc_1) it holds that*

1. *This member is $\mathbf{L}(\mathcal{Q}_1)$ -secure in the quantized domain; and*
2. *For any random variable X , and any member identified by \mathcal{Q}_2 (with encoder Enc_2), we have*

$$\tilde{\mathbf{H}}_\infty(\mathcal{Q}_2(X) | \text{Enc}_2(\mathcal{Q}_2(X))) - \tilde{\mathbf{H}}_\infty(\mathcal{Q}_1(X) | \text{Enc}_1(\mathcal{Q}_1(X))) \leq \mathbf{R}(\mathcal{Q}_1).$$

In other words, to measure the security of the family of schemes, we examine two aspects of the family. Firstly, we consider the entropy loss in the quantized domain for each member of the family. This is represented by the function \mathbf{L} , which serves as a measure of security when the quantizer is fixed. Secondly, given any

quantizer in the family, we consider the question: If we use another quantizer, how many more bits can be extracted? We call this the *relative entropy loss*, which is represented by the function \mathbf{R} .

We observe that for some sketch families, the relative entropy loss for any given member can be conveniently bounded by the size of the sketch generated by that member. We say that such sketch families are *well-formed*. More precisely, we have

Definition 5. *A quantization-based sketch family $(\mathcal{U}, \mathcal{S}, \mathcal{Q}, \mathcal{M}, \mathbf{E}, \mathbf{D})$ is well-formed if for any two members $(\mathcal{U}, \mathcal{S}, \mathcal{Q}_1, \mathcal{M}_1, \text{Enc}_1, \text{Dec}_1)$ and $(\mathcal{U}, \mathcal{S}, \mathcal{Q}_2, \mathcal{M}_2, \text{Enc}_2, \text{Dec}_2)$, it holds for any random variable X that*

$$\tilde{\mathbf{H}}_\infty(\mathcal{Q}_1(X) \mid \langle P_1, P_2 \rangle) = \tilde{\mathbf{H}}_\infty(\mathcal{Q}_2(X) \mid \langle P_1, P_2 \rangle) \quad (2)$$

where $P_1 = \text{Enc}_1(\mathcal{Q}_1(X))$ and $P_2 = \text{Enc}_2(\mathcal{Q}_2(X))$.

Theorem 1. *For any well-formed quantization-based sketch family, given any two members $(\mathcal{U}, \mathcal{S}, \mathcal{Q}_1, \mathcal{M}_1, \text{Enc}_1, \text{Dec}_1)$ and $(\mathcal{U}, \mathcal{S}, \mathcal{Q}_2, \mathcal{M}_2, \text{Enc}_2, \text{Dec}_2)$, it holds for any random variable X that*

$$\tilde{\mathbf{H}}_\infty(\mathcal{Q}_2(X) \mid P_2) - \tilde{\mathbf{H}}_\infty(\mathcal{Q}_1(X) \mid P_1) \leq |P_1|$$

where $P_1 = \text{Enc}_1(\mathcal{Q}_1(X))$ and $P_2 = \text{Enc}_2(\mathcal{Q}_2(X))$.

Proof: First, it is not difficult to show that for any random variables A, B and C , we have

$$\tilde{\mathbf{H}}_\infty(A \mid B) - |C| \leq \tilde{\mathbf{H}}_\infty(A \mid \langle B, C \rangle) \leq \tilde{\mathbf{H}}_\infty(A \mid B). \quad (3)$$

Let $\hat{X}_1 = \mathcal{Q}_1(X)$ and $\hat{X}_2 = \mathcal{Q}_2(X)$. Since the sketch family is well-formed,

$$\tilde{\mathbf{H}}_\infty(\hat{X}_1 \mid \langle P_1, P_2 \rangle) = \tilde{\mathbf{H}}_\infty(\hat{X}_2 \mid \langle P_1, P_2 \rangle). \quad (4)$$

Substituting B by P_1 , C by P_2 , and A by \hat{X}_1 and \hat{X}_2 respectively in (3), we have

$$\begin{aligned} \tilde{\mathbf{H}}_\infty(\hat{X}_2 \mid P_2) - |P_1| &\leq \tilde{\mathbf{H}}_\infty(\hat{X}_2 \mid \langle P_1, P_2 \rangle) \\ &= \tilde{\mathbf{H}}_\infty(\hat{X}_1 \mid \langle P_1, P_2 \rangle) \leq \tilde{\mathbf{H}}_\infty(\hat{X}_1 \mid P_1). \end{aligned} \quad (5)$$

□

5 A General Scheme for Biometric Templates

We observe that many biometric templates can be represented as a sequence of points in some bounded continuous domain. There are two types of noise that can occur. The first noise, *white noise*, perturbs each points by a small distance, and the second noise, *replacement noise*, replaces some points by different points.

Without loss of generality, we assume that each biometric template X can be written as a sequence $X = \langle x_1, x_2, \dots, x_n \rangle$, where each $x_i \in \mathbb{R}$ and $0 \leq x_i < 1$. In other words, $X \in \mathcal{U} = [0, 1]^n$. For each pair of biometric templates X and Y , we say that $(X, Y) \in \mathcal{S}$ if there exists a subset C of $\{1, \dots, n\}$, such that $|C| \geq n - t$ for some threshold t , and for every $i \in C$, it holds that $|x_i - y_i| < \delta$, for some threshold δ .

Similar to the two-part approach in [5], we construct the sketch in two parts. The first part, the *white noise sketch*, handles the white noise in the noisy data, and the second part, the *replacement noise sketch*, corrects the replacement noise. We will concentrate on the white noise sketch in this paper, and the replacement noise sketch can be implemented using a known secure sketch scheme for set difference (e.g., that in [7,3]).

5.1 Proposed Quantization-Based Sketch Family

Each member of the family is parameterized by a λ such that $\lambda \in \mathbb{R}$ and $0 < \lambda \leq \delta$.

Quantizer \mathcal{Q}_λ . Each quantizer \mathcal{Q}_λ in \mathbf{Q} is a scalar quantizer with step size $\lambda \in \mathbb{R}$. For each $x \in \mathcal{U}$, $\mathcal{Q}_\lambda(x) = \hat{x}$ if and only if $\lambda\hat{x} \leq x < \lambda(\hat{x} + 1)$, and the quantization of X is defined as $\hat{X} = \mathcal{Q}_\lambda(X) \triangleq \langle \mathcal{Q}_\lambda(x_1), \dots, \mathcal{Q}_\lambda(x_n) \rangle$. The corresponding quantized domain is thus $\mathcal{M}_\lambda = [0, \lceil \frac{1}{\lambda} \rceil]^n$. The encoders and the decoders work only on the quantized domain. The white noise appeared in the quantized domain is of level $\hat{\delta}_\lambda = \lceil \delta/\lambda \rceil$. In other words, under white noise, a point \hat{x} in the quantized domain can be shifted by a distance of at most $\hat{\delta}_\lambda$. Let us denote $\Delta_\lambda \triangleq 2\hat{\delta}_\lambda + 1$.

Codebook \mathcal{C}_λ . Furthermore, for each quantized domain \mathcal{M}_λ we consider a *codebook* \mathcal{C}_λ , where every codeword $c \in \mathcal{C}_\lambda$ has the form $c = k\Delta_\lambda$ for some non-negative integer k . We use $\mathcal{C}_\lambda(\cdot)$ to denote the function such that given a quantized point \hat{x} , it returns a value $c = \mathcal{C}_\lambda(\hat{x})$ such that $|\hat{x} - c| \leq \hat{\delta}_\lambda$. That is, the functions finds the unique codeword c that is nearest to \hat{x} in the codebook.

Encoder Enc_λ . Given a quantized $\hat{X} \in \mathcal{M}_\lambda$, the encoder Enc_λ does the following.

1. For each $\hat{x}_i \in \hat{X}$, compute $c_i = \mathcal{C}_\lambda(\hat{x}_i)$;
2. Output $P = \text{Enc}_\lambda(\hat{X}) = \langle d_1, \dots, d_n \rangle$, where $d_i = \hat{x}_i - c_i$ for $1 \leq i \leq n$.

In other words, for every \hat{x}_i , the encoder outputs the distance of \hat{x}_i from its nearest codeword in the codebook \mathcal{C}_λ .

Decoder Dec_λ . For a corrupted template Y , it is first quantized by $\hat{Y} = \mathcal{Q}_\lambda(Y)$. Given $P = \langle d_1, \dots, d_n \rangle$ and $\hat{Y} = \langle \hat{y}_1, \dots, \hat{y}_n \rangle$, and the decoder Dec_λ does the following.

1. For each $\hat{y}_i \in \hat{Y}$, compute $c_i = \mathcal{C}_\lambda(\hat{y}_i - d_i)$;
2. Output $\hat{X} = \text{Dec}_\lambda(\hat{Y}) = \langle c_1 + d_1, \dots, c_n + d_n \rangle$.

In other words, the decoder shifts every \hat{y}_i by d_i , maps it to the nearest codeword in \mathcal{C}_λ , and shifts it back by the same distance.

5.2 Security Analysis

For each member of the sketch family with parameter λ , the difference d_i between \hat{x}_i and p_i ranges from $-\hat{\delta}_\lambda$ to $\hat{\delta}_\lambda$. Intuitively, $\log \Delta_\lambda$ bits are sufficient and necessary to describe the white noise in the quantized domain (recall that $\Delta_\lambda = 2\hat{\delta}_\lambda + 1 = 2\lceil \frac{\delta}{\lambda} \rceil + 1$). Hence, we have

Lemma 2. *The quantization-based sketch scheme $(\mathcal{U}, \mathcal{S}, \mathcal{Q}_\lambda, \mathcal{M}_\lambda, \text{Enc}_\lambda, \text{Dec}_\lambda)$ is $(n \log \Delta_\lambda)$ -secure in the quantized domain.*

Proof: Note that the size of each d_i generated in the second step of the encoder is $\log \Delta_\lambda$. Hence the total size of the sketch is $n \log \Delta_\lambda$. Therefore, the entropy loss of the sketch P is at most $n \log \Delta_\lambda$ by Equation (1). \square

It is not difficult to see that the above bound is tight. For example, when each \hat{x} is uniformly distributed in the quantized domain, the min-entropy of each \hat{x} after quantization would be $\log \lceil \frac{1}{\lambda} \rceil$, and the average min-entropy of \hat{x} given P would be at most $\log |\mathcal{C}_\lambda| = \log \lceil \frac{1}{\lambda} \rceil - \log \Delta_\lambda$.

Now we consider the relative entropy loss. First of all, we observe that the proposed sketch family is well-formed according to Definition 5.

Lemma 3. *The quantization-based sketch family defined in Section 5.1 is well-formed.*

Proof: We consider any two members in the sketch family. The first is identified by \mathcal{Q}_{λ_1} with step size λ_1 , and the second is identified by \mathcal{Q}_{λ_2} with step size λ_2 .

For any point $x \in X$, let $\hat{x}_1 = \mathcal{Q}_{\lambda_1}(x)$. Recall that during encoding, a code-word is computed as $c_1 = \mathcal{C}_{\lambda_1}(\hat{x}_1)$, and the difference $d_1 = \hat{x}_1 - c_1$ is put into the sketch. Similarly, let $\hat{x}_2 = \mathcal{Q}_{\lambda_2}(x)$, $c_2 = \mathcal{C}_{\lambda_2}(\hat{x}_2)$ and $d_2 = \hat{x}_2 - c_2$.

Since $\lambda_1 \leq \delta$ and $\lambda_2 \leq \delta$, it is easy to see that if d_1, d_2 and \hat{x}_1 is known, we can compute \hat{x}_2 deterministically. Similarly, given d_1, d_2 and \hat{x}_2 , \hat{x}_1 can also be determined. Thus, we have

$$\tilde{\mathbf{H}}_\infty(\hat{x}_1 \mid \langle d_1, d_2 \rangle) = \tilde{\mathbf{H}}_\infty(\langle \hat{x}_1, \hat{x}_2 \rangle \mid \langle d_1, d_2 \rangle) = \tilde{\mathbf{H}}_\infty(\hat{x}_2 \mid \langle d_1, d_2 \rangle). \quad (6)$$

The same arguments can be applied to all the points in X . Hence, let $P_1 = \text{Enc}_{\lambda_1}(X)$ and $P_2 = \text{Enc}_{\lambda_2}(X)$, we have

$$\tilde{\mathbf{H}}_\infty(\hat{X}_1 \mid \langle P_1, P_2 \rangle) = \tilde{\mathbf{H}}_\infty(\langle \hat{X}_1, \hat{X}_2 \rangle \mid \langle P_1, P_2 \rangle) = \tilde{\mathbf{H}}_\infty(\hat{X}_2 \mid \langle P_1, P_2 \rangle). \quad (7)$$

That is, the proposed sketch family is well-formed. \square

By combining Theorem 1 and Lemma 3, and considering that for the member of the sketch family identified by \mathcal{Q}_{λ_1} with step size λ_1 , the size of the sketch $|P_1| = n(\log \Delta_{\lambda_1})$, we have the following lemma.

Lemma 4. *For the quantization-based sketch family defined in Section 5.1, given any member identified by \mathcal{Q}_{λ_1} with step size λ_1 and encoder Enc_{λ_1} it holds that, for*

every random variable $X \in \mathcal{U}$ and any member identified by \mathcal{Q}_{λ_2} with step size λ_2 and encoder Enc_{λ_2} , we have

$$\tilde{\mathbf{H}}_{\infty}(\mathcal{Q}_{\lambda_2}(X) \mid \text{Enc}_{\lambda_2}(\mathcal{Q}_{\lambda_2}(X))) - \tilde{\mathbf{H}}_{\infty}(\mathcal{Q}_{\lambda_1}(X) \mid \text{Enc}_{\lambda_1}(\mathcal{Q}_{\lambda_1}(X))) \leq n(\log \Delta_{\lambda_1}).$$

In other words, the relative entropy loss is at most $n(\log \Delta_{\lambda_1})$ for \mathcal{Q}_{λ_1} .

Not only the above is a worst case bound, we can show that the worst case can indeed happen.

Lemma 5. *The relative entropy loss in Lemma 4 is tight for sufficiently small δ .*

Proof: For any given λ_1 , we find a λ_2 such that it is possible to find $\Delta_{\lambda_1} \triangleq (2\lceil \delta/\lambda_1 \rceil + 1)$ points $W = \{w_0, \dots, w_{\Delta_{\lambda_1}-1}\}$ such that $\mathcal{Q}_{\lambda_1}(w_i) - \mathcal{C}_{\lambda_1}(\mathcal{Q}_{\lambda_1}(w_1)) = i - \lceil \delta/\lambda_1 \rceil$, and $\mathcal{C}_{\lambda_2}(w_i) = c_i$ for some codeword $c_i \in \mathcal{C}_{\lambda_2}$. In other words, we want to find points such that each of them would generate a different d_i in the final sketch with \mathcal{Q}_{λ_1} , but would generate exactly the same number (i.e., 0) in the sketch when \mathcal{Q}_{λ_2} is used. Note that when δ is sufficiently small, there would be sufficiently many codewords in \mathcal{C}_{λ_1} , and it is always possible to find such λ_2 (e.g., $\lambda_2 = \lambda_1/2$).

When each $x \in X$ is uniformly distributed over W , we can see that the sketch from the scheme identified by \mathcal{Q}_{λ_1} would reveal all information about X , but in the case of \mathcal{Q}_{λ_2} , the left-over entropy would be exactly $\log \Delta_{\lambda_1}$. \square

Therefore, combining lemmas 2, 4 and 5 we have

Theorem 6. *The quantization-based sketch family defined in Section 5.1 is (\mathbf{L}, \mathbf{R}) -secure where for each member in the family identified by \mathcal{Q}_{λ} with step size λ , where $\mathbf{L}(\mathcal{Q}_{\lambda}) = \mathbf{R}(\mathcal{Q}_{\lambda}) = n \log \Delta_{\lambda}$. Furthermore, the bounds are tight.*

For example, if $\lambda = \delta$, we would have $\mathbf{L}(\mathcal{Q}_{\lambda}) = \mathbf{R}(\mathcal{Q}_{\lambda}) = n(\log 3)$. Note that although decreasing λ might give a larger left-over entropy, this is not guaranteed. In fact, if we use a $\lambda' < \lambda$, by applying the above theorem on $\mathcal{Q}_{\lambda'}$, we can see that it may result in a smaller left-over entropy than using \mathcal{Q}_{λ} (e.g., consider the example in the proof of Lemma 5).

5.3 A Special Case

We further study a special case when each point $x \in X$ is independently and uniformly distributed over $[0, 1)$. We further assume that $1/\delta$ is an integer, and the family of schemes only consists of members with step size λ such that $1/\lambda$ is an integer that is a multiple of Δ_{λ} . This additional assumption is only for the convenience of the analysis, and would not make too much difference in practice.

In this case, the entropy loss in the quantized domain for the member identified by \mathcal{Q}_{λ} with step size λ would be exactly $n(\log \Delta_{\lambda})$, which shows that Lemma 2 is tight. Moreover, it is interesting that the relative entropy loss in this case can be bounded by a constant.

Corollary 7. *When each $x \in X$ is independently and uniformly distributed, the quantization-based sketch family defined in Section 5.1 is (\mathbf{L}, \mathbf{R}) -secure where for each member in the family identified by \mathcal{Q}_λ with step size λ , where $\mathbf{L}(\mathcal{Q}_\lambda) = n(\log \Delta_\lambda)$, and $\mathbf{R}(\mathcal{Q}_\lambda) = n \log(1 + \frac{\lambda}{2\delta}) \leq n \log(3/2)$.*

Proof: The claim $\mathbf{L}(\mathcal{Q}_\lambda) = n(\log \Delta_\lambda)$ follows directly from Lemma 2, so we only focus on \mathbf{R} . Consider two members of the family identified by \mathcal{Q}_{λ_1} and \mathcal{Q}_{λ_2} respectively. Without loss of generality, we assume $\lambda_1 > \lambda_2$. Consider any $x \in X$, let $\hat{x}_1 = \mathcal{Q}_{\lambda_1}(x)$, $c_1 = \mathcal{C}_{\lambda_1}(\hat{x}_1)$. Similarly we define $\hat{x}_2 = \mathcal{Q}_{\lambda_2}(x)$ and $c_2 = \mathcal{C}_{\lambda_2}(\hat{x}_2)$. Hence, the min-entropy in the quantized domain would be $\log(1/\lambda_1)$ and $\log(1/\lambda_2)$ respectively.

Clearly, c_1 and c_2 are also uniformly distributed over \mathcal{C}_{λ_1} and \mathcal{C}_{λ_2} respectively, and do not depend on d_1 and d_2 . Hence, the left-over entropy for these two members would be $\log(|\mathcal{C}_{\lambda_1}|) = \log \frac{1}{\lambda_1 + 2\delta}$ and $\log(|\mathcal{C}_{\lambda_2}|) = \log \frac{1}{\lambda_2 + 2\delta}$ respectively. Furthermore, recall that $0 < \lambda_2 < \lambda_1 \leq \delta$, and the difference between these two quantities can be bounded as

$$\log(|\mathcal{C}_{\lambda_2}|) - \log(|\mathcal{C}_{\lambda_1}|) = \log \frac{\lambda_1 + 2\delta}{\lambda_2 + 2\delta} < \log(1 + \frac{\lambda_1}{2\delta}) \leq \log \frac{3}{2}.$$

Therefore, the relative entropy loss is bounded by $n \log(3/2)$ as claimed. \square

5.4 Remarks

Choosing the step size λ . We can view the step size λ as a measure of the precision of \hat{X} . Since the white noise in the continuous domain is fixed at δ , when λ becomes smaller, the corresponding white noise in the quantized domain would increase, and vice versa. That is intuitively why it is not possible to obtain much more left-over entropy by simply having X represented in a higher precision. In fact, it is not difficult to show that there are certain distributions of X such that a smaller step size would reveal more information. Furthermore, the scheme can be more efficient if we use a relatively larger step size, since we would need fewer bits to represent both X and the white noise in the quantized domain. If we use the same quantizer for both encoding and decoding, the simplest form of white noise in the quantized domain can be achieved when $\lambda = \delta$, where a quantized \hat{x} can be either left unchanged, or shifted by 1. In this case, from Theorem 6, we can get at most $n \log 3$ additional bits if we choose other $\lambda' < \delta$. If X is uniformly distributed, the increment is at most $n \log(3/2)$ by Corollary 7.

When $\lambda > \delta$, the form of white noise in the quantized domain would remain unchanged, but we may lose too much information about X due to the large quantization step, which may result in a much lower left-over entropy. Therefore, it is not desirable to have a step size larger than δ in general. If different quantizers are used during encoding and decoding, with large step size (e.g., 2δ), it is possible to reduce the white noise in the quantized domain to a special 0-1 noise, under which an \hat{x} is either left unchanged or shifted to $\hat{x} + 1$, as observed in [4]. Nevertheless, this strategy may give lower left-over entropy.

Handling replacement noise. After the white noise has been corrected, an existing scheme for set difference can be applied in the quantized domain to correct the replacement noise. There are known schemes that can achieve entropy loss of $O(t \log \lceil \frac{1}{\lambda} \rceil)$ with small leading constant, such as those in [7,3]. Although the replacement noise is not considered for the face biometrics that we study in Section 6, it may need to be addressed for other biometric templates (e.g., iris patterns [9]).

Extension to higher dimensions. It is straightforward to extend our scheme to higher dimensions, where each $x \in X$ is a point in some d -dimensional space. For example, we can apply a scalar quantizer on each coordinate of every point, and let the distance of two points in d -dimensional space be measured by max-norm (i.e., the maximum distance in all dimensions). The entropy loss of the resulting scheme would be d times that in the current construction for 1-D points. If there is no replacement noise, we could also expand the n points in d -dimensional space into nd points in 1-D and apply the proposed scheme.

The choice of the sketch family. It is important to note that even if a quantization-based sketch family is well-formed, it does not guarantee the existence of a “good” quantizer in that family. Nevertheless, it does allow us to evaluate any given member in the family with respect to the “optimal” member in the family. We consider it a challenging open problem to find a general algorithm to find the optimal quantizer among all possible quantizers, given certain practical constraints (e.g., the smallest possible quantization step and the distribution of X).

6 A Concrete Construction for Face Biometrics

Face images, especially those taken from a controlled environment, can be used as the basis of identity verification. Here we follow the techniques employed in [17] and make use of the *singular value decomposition* (SVD) of the face images for verification, which is a well-known strategy in the face recognition literature (such as [10,6]). Given a face image A of size $M \times N$, we can always find matrices U , Σ and V such that $A = U\Sigma V^T$, where Σ is an $M \times N$ matrix with $\min(M, N)$ non-zero elements ordered according to their significance. As noted in [17], some (say, n) most significant coefficients of Σ contain significant identity information of the individual. Typically n is chosen such that the sum of these n coefficients is more than, say, 98% of the sum of all the coefficients.

In [17], the biometric template of an individual is obtained as follows. First, we take a few face images, compute the SVD, and obtain the minimum \min_i and maximum \max_i of the i -th significant coefficient, for $1 \leq i \leq n$, where n is chosen to be 20. The mean value $a_i = (\max_i + \min_i)/2$ is then taken as a point in the template. When a new face image is presented for verification, its SVD is computed, and if for $1 \leq i \leq n$, the i -th significant coefficient is sufficiently close to a_i , it is considered as authenticated. The scheme in [17] is applied to face images from the Essex Faces94 Database [16], which contains 152 faces with 20 images for each face (24bit color JPEG). Twelve images per

face are randomly chosen to compute the templates, and the rest 8 are used for testing. The experiments show that when the false accept rate is 0.005, the false reject rate is less than 0.045.

To apply our sketch scheme, for each coefficient, we further compute the minimum \min and the maximum \max of all the templates in the database (assuming that the number of templates is large). Hence, we can compute our biometric template X as a sequence of n points, where the i -th point $x_i = \frac{a_i - \min}{\max - \min}$. We set the noise level $\delta_i = \frac{k(\max x_i - a_i)}{\max - \min}$ for some constant $k \geq 1$. In this way, each point x_i will be between 0 and 1 so that our scheme can be applied. There is a difference, however, that we have a different δ_i for each point, which we have to put as part of the sketch. Nevertheless, our analysis on the entropy loss can be easily adapted to this case, and the difference here will not affect the security of the scheme. Here we choose $\lambda_i = \delta_i$ for all $1 \leq i \leq n$.

In this way, the sketch produced by our proposed scheme, would be the tuple

$$P = (\min, \max, \lambda_1, \dots, \lambda_n, \hat{x}_1 - \mathcal{C}_{\lambda_1}(\hat{x}_1), \dots, \hat{x}_n - \mathcal{C}_{\lambda_n}(\hat{x}_n))$$

where $\hat{x}_i = \mathcal{Q}_{\lambda_i}(x_i)$ for $1 \leq i \leq n$. By applying the arguments in Theorem 6 and Corollary 7 to each point in X , we have

Corollary 8. *The entropy loss in the quantized domain for the aforementioned scheme is at most $n \log 3$. Let m be the left-over entropy. When $\lambda_i < \delta_i$ for any i , $1 \leq i \leq n$, let the left-over entropy be m' . We have $m' - m \leq n \log 3$. If all points are uniformly distributed, we have $m' - m \leq n \log(3/2)$.*

When $n = 20$, the above bounds are approximately 31.7 and 11.7 respectively.

References

1. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In *Eurocrypt*, 2005.
2. Xavier Boyen. Reusable cryptographic fuzzy extractors. In *ACM CCS*, pages 82–91, Washington DC, USA, 2004. ACM Press.
3. Ee-Chien Chang, Vadym Fedyukovych, and Qiming Li. Secure sketch for multi-set difference. Cryptology ePrint Archive, Report 2006/090, 2006. <http://eprint.iacr.org/>.
4. Ee-Chien Chang and Qiming Li. Small secure sketch for point-set difference. Cryptology ePrint Archive, Report 2005/145, 2005. <http://eprint.iacr.org/>.
5. Ee-Chien Chang and Qiming Li. Hiding secret points amidst chaff. In *Eurocrypt*, volume 4004 of *LNCS*, pages 59–72, 2006.
6. Yong-Qing Cheng. Human face recognition method based on the statistical model of small sample size. In *SPIE Proc. Intell. Robot and Compu. Vision*, pages 85–95, 1991.
7. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Eurocrypt*, volume 3027 of *LNCS*, pages 523–540. Springer-Verlag, 2004.
8. F. Hao and C.W. Chan. Private key generation from on-line handwritten signatures. *Information Management and Computer Security*, 10(2), 2002.

9. Feng Hao, Ross Anderson, and John Daugman. Combining cryptography with biometrics effectively. Technical Report UCAM-CL-TR-640, University of Cambridge, 2005.
10. Z. Hong. Algebraic feature extraction of image for recognition. *Pattern Recognition*, 24:211–219, 1991.
11. Ari Juels and Madhu Sudan. A fuzzy vault scheme. In *IEEE Intl. Symp. on Information Theory*, 2002.
12. Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *ACM CCS*, pages 28–36, 1999.
13. J.-P. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *AVBPA*, pages 393–402, 2003.
14. Ueli Maurer and Stefan Wolf. Information-theoretic key agreement: From weak to strong secrecy for free. In *Eurocrypt*, 2000.
15. F. Monrose, M.K. Reiter, Q. Li, and S. Wetzels. Cryptographic key generation from voice. In *IEEE Symp. on Security and Privacy*, 2001.
16. Libor Spacek. The essex faces94 database. <http://cswww.essex.ac.uk/mv/allfaces/>.
17. Y. Sutcu, T. Sencar, and N. Memon. A secure biometric authentication scheme based on robust hashing. In *ACM MM-SEC Workshop*, 2005.
18. P. Tuyls, A.H.M. Akkermans, T.A.M. Kevenaar, G.J. Schrijen, A.M. Bazen, and R.N.J. Veldhuis. Practical biometric authentication with template protection. In *AVBPA*, pages 436–446, 2005.
19. P. Tuyls and J. Goseling. Capacity and examples of template-protecting biometric authentication systems. In *ECCV Workshop BioAW*, pages 158–170, 2004.
20. Shenglin Yang and Ingrid Verbauwhede. Automatic secure fingerprint verification system based on fuzzy vault scheme. In *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 609–612, 2005.

The 2-Adic CM Method for Genus 2 Curves with Application to Cryptography

P. Gaudry^{1,2}, T. Houtmann², D. Kohel³, C. Ritzenthaler⁴, and A. Weng²

¹ LORIA - Projet SPACES

Campus Scientifique - BP 239, 54506 Vandoeuvre-ls-Nancy Cedex France

² Laboratoire d'Informatique (LIX)

École polytechnique, 91128 Palaiseau Cedex France

³ School of Mathematics and Statistics

The University of Sydney, NSW 2006 Australia

⁴ Institut de Mathématiques de Luminy

163 Avenue de Luminy, Case 907, 13288 Marseille Cedex 9 France

Abstract. The complex multiplication (CM) method for genus 2 is currently the most efficient way of generating genus 2 hyperelliptic curves defined over large prime fields and suitable for cryptography. Since low class number might be seen as a potential threat, it is of interest to push the method as far as possible. We have thus designed a new algorithm for the construction of CM invariants of genus 2 curves, using 2-adic lifting of an input curve over a small finite field. This provides a numerically stable alternative to the complex analytic method in the first phase of the CM method for genus 2. As an example we compute an irreducible factor of the Igusa class polynomial system for the quartic CM field $\mathbb{Q}(i\sqrt{75 + 12\sqrt{17}})$, whose class number is 50. We also introduce a new representation to describe the CM curves: a set of polynomials in (j_1, j_2, j_3) which vanish on the precise set of triples which are the Igusa invariants of curves whose Jacobians have CM by a prescribed field. The new representation provides a speedup in the second phase, which uses Mestre's algorithm to construct a genus 2 Jacobian of prime order over a large prime field for use in cryptography.

1 Introduction

In the late 1980's, Koblitz proposed the use of hyperelliptic curves in cryptography. Since then, significant progress has been made in turning this idea into practice, and currently genus two cryptosystems present the same security benefits as elliptic curves, together with potential benefits in terms of performance and new protocols [31,2,17,22].

The efficient generation of genus two groups of prime or nearly prime order over finite fields of large characteristic, however, remains an important issue. Random curve generation in characteristic 2 is amenable to efficient versions of Kedlaya's algorithm or Mestre's AGM algorithm. In contrast, over large prime fields the latest records for point counting (see [18]) still require about a week's

computation time for each curve. In this case, the complex multiplication method currently provides the only efficient approach to cryptographic curve construction. For genus one, several authors have introduced improvements to the CM method using p -adic lifting [13,7,6,24]. Our article generalizes such work to the case of genus two. Furthermore, in the past few years, the elliptic CM method has gained new interest as the key tool for building curves with a special structure, in particular curves with a computable bilinear map [29]. Similar constructions in genus two will also require explicit CM methods.

The first phase of the CM method constructs the *Igusa class polynomials* for CM genus two curves, which determine the triples (j_1, j_2, j_3) of invariants of curves whose Jacobians have prescribed endomorphism ring. These polynomials are determined by complex analytic techniques, or, in this work, by p -adic analytic construction. After solving for the roots of these polynomials over a chosen finite field of large characteristic, the algorithm of Mestre [28] allows one to construct a model of the curve for which the group order of its Jacobian has been previously determined to be prime or nearly prime. In this article, we extend the computational limit for Igusa class polynomials in genus two, addressing concerns that a CM field of low class number might give rise to weak curves in a cryptographic protocol.

Our first contribution is to use a 2-adic lifting method in place of the classical floating point complex approach. We start with a binary curve over a field small enough so that point counting is possible using naive methods. We determine not only the number of points but also the endomorphism ring of the Jacobian and therefore the CM field K associated to it. By computing the canonical 2-adic lift with sufficiently high precision we are able to get the class polynomials which we recognize as polynomials over the rationals. This bypasses the costly step of evaluating theta functions. We also introduce a simple representation of the ideal of CM invariants in terms of univariate polynomials. Prior authors focused on finding the degree h_K^* minimal polynomials $H_1(X)$, $H_2(X)$, and $H_3(X)$ of the invariants j_1 , j_2 , and j_3 . However in the second phase of the CM method, this requires a combinatorial match of $h_K^*{}^3$ roots to find one of h_K^* valid triples, when constructing a CM curve. For those small values of h_K^* previously attainable, this was not particularly onerous, but with our 2-adic method, our largest examples computed have reached $h_K^* = 100$, for which this combinatorial matching problem is undesirable.

Our *Magma* and *C* implementation of the 2-adic CM method allow us to compute a degree 50 irreducible factor of Igusa class polynomials for the quartic CM field $K = \mathbb{Q}(i\sqrt{75 + 12\sqrt{17}})$. The class number of K is 50 and the Igusa class polynomials for K have degree $h_K^* = 100$.

The paper is organized as follows. In section 2 we introduce the mathematical objects we need to explain the 2-adic CM method and the generation of hyper-elliptic curves suitable for cryptography. In section 3 we deal with Igusa class polynomials, our new representation of the ideal of invariants. In section 4 we give details about the 2-adic CM method. In section 5 we analyze its complexity and compare it with previous methods [35,40,9,16].

2 Mathematical Background

In this section, we briefly present the mathematical tools that we need. The first part deals with complex multiplication theory. We give theoretical results applied to our genus two case. Then we recall Lubin-Serre-Tate theorem for genus two and finally we deal with the reduction of the variety of j -invariants.

2.1 Complex Multiplication Theory

We begin with some definitions and results from the theory of complex multiplication (see [33] for further details). The central notion is that of a *CM field*, defined to be a totally imaginary quadratic extension K of a totally real number field K_0 .

For the study of genus two curves we will be interested in quartic CM fields K . We define a *type* of such a field as a pair of non-conjugate embeddings $\Phi = (\phi_1, \phi_2)$ of K in \mathbb{C} . If I is an ideal in the ring of integers \mathcal{O}_K of K , we consider $\Phi(I) = \{(\phi_1(\alpha), \phi_2(\alpha)) \in \mathbb{C}^2, \alpha \in I\}$. The set $\Phi(I)$ is a lattice in \mathbb{C}^2 and $\mathbb{C}^2/\Phi(I)$ is an abelian variety A such that $K \subset \text{End}(A) \otimes \mathbb{Q}$. We furthermore make the following restrictions:

1. We assume that K is cyclic or non-Galois. The abelian variety A (for which $\text{End}(A) \otimes \mathbb{Q} = K$) is then absolutely simple. This is a good condition for cryptographic applications since we want $\#A(\mathbb{F}_q)$ to be almost prime.
2. We assume that $h_{K_0} = 1$, which implies that the abelian surface A has a principal polarization. As A is absolutely simple, it follows there exists a genus two curve \mathcal{C} such that $A = \text{Jac}(\mathcal{C})$.
3. We assume moreover that $\text{End}(\text{Jac}(\mathcal{C})) = \mathcal{O}_K$. The above conditions imply $\text{End}(\text{Jac}(\mathcal{C})) \subseteq \mathcal{O}_K$, but for sake of simplicity of both theory and computations, we restrict to the case where this inclusion is an equality. This requires us to address the issue of testing effectively this hypothesis for a given curve \mathcal{C} , but we will not treat these algorithms in this article (see however [16]).

Definition 1. *Let \mathcal{C} be a hyperelliptic curve of genus two and K a quartic CM field. We say that \mathcal{C} has complex multiplication by \mathcal{O}_K if the endomorphism ring of the Jacobian of the curve is isomorphic to the ring of integers \mathcal{O}_K of K .*

Example 1. As an example we consider $K = \mathbb{Q}(i\sqrt{2 + \sqrt{2}})$. The real subfield of K is $\mathbb{Q}(\sqrt{2})$ since $(i\sqrt{2 + \sqrt{2}})^2 + 2 = -\sqrt{2}$. Then there exists a curve defined over \mathbb{Q} with model $y^2 = -x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1$, whose Jacobian has endomorphism ring \mathcal{O}_K . Further details on this example can be found in [38] or [35].

We first recall basic notions of CM theory in genus one, for which we refer to [3]. We begin with a positive squarefree integer D , and compute the class group of $K = \mathbb{Q}(i\sqrt{D})$, which we denote by Cl_K . For complex numbers $(\tau_i)_{i \in [1, h_K]}$, representing the classes in Cl_K , we associate an elliptic curve with

period lattice $\mathbb{Z} + \tau_i\mathbb{Z}$. Finally we compute the j -invariant $j_i = j(\tau_i)$ using η -functions and recover the classical Hilbert class polynomial from the definition $H(X) = \prod_{i=1}^{h_K} (X - j_i) \in \mathbb{Z}[X]$, as a monic polynomial over the integers.

The analogous theory for genus two presents several additional technical challenges. The first question is to determine how many isomorphism classes of CM curves are associated to a CM order \mathcal{O}_K . We denote this number by h_K^* . In genus one, this number equals the class number h_K , but in higher genus there is no longer a one-to-one correspondence between the ideal classes and the principally polarized abelian surfaces with endomorphism ring \mathcal{O}_K , each of which gives rise to an isomorphism class of CM curves. However, for a quartic CM field K with real subfield of class number one, we can make the following statement.

Theorem 1. *Let K be a quartic CM field with real quadratic subfield K_0 of class number 1. If K is cyclic over \mathbb{Q} then there are h_K isomorphism classes and if K is not normal over \mathbb{Q} then there are $2h_K$ isomorphism classes with h_K classes associated to each CM type.*

Remark 1. The Cohen-Lenstra heuristics [11] predict that the class number of the real quadratic field K_0 has class number 1 with density greater than $3/4$ so this is expected to apply to this proportion of all quartic CM fields.

The above theorem establishes the degree of the Igusa class polynomials, which vanish on the triples of the CM Igusa invariants (j_1, j_2, j_3) . Once their degree is known, we can apply a construction as in the genus 1 CM method for the classical complex CM method. Beginning from a quartic CM field K , we compute the class group of K over \mathbb{Q} , and find a representative of each class. Here the representatives are 2×2 matrices called *period matrices* which can be computed from a set of representatives of the class group of K and a fundamental unit of K_0 . We refer to [40] for the exact construction of these period matrices $(\Omega_i)_{1 \leq i \leq h_K^*}$.

Evaluating theta functions at the Ω_i allows to recover the j -invariants $(j_1^{(i)}, j_2^{(i)}, j_3^{(i)})_i$ of the CM curves and joining the j -invariants together gives us the Igusa class polynomials described in [35] or in [40] as

$$H_1 = \prod_{i=1}^{h_K^*} (X - j_1^{(i)}), \quad H_2 = \prod_{i=1}^{h_K^*} (X - j_2^{(i)}), \quad H_3 = \prod_{i=1}^{h_K^*} (X - j_3^{(i)}).$$

For the purposes of 2-adic lifting we may use normalized invariants j_1, j_2 , and j_3 , defined in terms of the Igusa-Clebsch invariants A, B, C, D (denoted A', B', C', D' in Mestre [28]), by $j_1 = A^5/8D, j_2 = 2A^3B/D, j_3 = 8A^2C/D$.

2.2 The Lubin-Serre-Tate Theorem for Genus Two

In 1964, Lubin, Serre and Tate [25] proved the existence of the canonical lift of an ordinary abelian variety and gave a way of computing this lift for elliptic curves, extending a result of Deuring [14]. Denote by \mathbb{Q}_p the field of p -adic numbers, and

by \mathbb{Q}_{p^d} the unique unramified extension of degree d , and by \mathbb{Z}_p or \mathbb{Z}_{p^d} their respective rings of integers (see e.g. [4] or [21] for background). The fundamental property of the canonical lift $A^\dagger/\mathbb{Z}_{p^d}$ of an ordinary abelian variety A/\mathbb{F}_{p^d} is that $\text{End}(A^\dagger) \cong \text{End}(A)$. Moreover, A^\dagger is actually defined over $\overline{\mathbb{Q}}$. Thus if we can find a curve over \mathbb{F}_{p^d} whose Jacobian is ordinary and has complex multiplication by the ring of integers of a quartic CM field K , we theoretically obtain a curve over $\overline{\mathbb{Q}}$ with complex multiplication by \mathcal{O}_K . In the article, p is fixed to 2 and the CM-curves over \mathbb{F}_{2^d} whose Jacobian is ordinary are not rare and can be found easily.

To perform this method explicitly, we require a constructive formulation of the existence theorem for the canonical lift. In genus 1, this is the following theorem (see [39]).

Theorem 2. *Let p be a prime number and d an integer greater than 2. Let \bar{E} be an ordinary elliptic curve over \mathbb{F}_{p^d} with j -invariant $j(\bar{E}) \in \mathbb{F}_{p^d} \setminus \mathbb{F}_{p^2}$. Denote by σ the Frobenius automorphism of \mathbb{Z}_{p^d} and by $\Phi_p(X, Y)$ the p -th modular polynomial. Then the system of equations*

$$\Phi_p(X, X^\sigma) = 0 \text{ and } X \equiv j(\bar{E}) \pmod{p},$$

has a unique solution $J \in \mathbb{Z}_{p^d}$, which is the j -invariant of the canonical lift E of \bar{E} (defined up to isomorphism).

Generalization to genus two is easier if one speaks about isogeny instead of modular equations:

Theorem 3. *Let $\bar{\mathcal{C}}$ be an ordinary hyperelliptic curve of genus two over \mathbb{F}_{p^d} . Then there exists a hyperelliptic curve \mathcal{C} of genus two defined over \mathbb{Q}_{p^d} that is a canonical lift of $\bar{\mathcal{C}}$ (in the sense that the endomorphism ring of the Jacobian is preserved) and furthermore there exists a (p, p) -isogeny between $\text{Jac}(\mathcal{C})$ and $\text{Jac}(\mathcal{C}^\sigma)$ that reduces to the Frobenius map from $\text{Jac}(\bar{\mathcal{C}})$ to its conjugate.*

In the case where $p = 2$, the Richelot isogeny [5] provides explicit formulae that allow us to translate this theorem into a set of equations that must be satisfied by the defining equation of the canonical lift. A Newton-like process due to Harley is then used to solve it (more details are given in Section 4.1).

General results on the convergence of the Newton process for the AGM is given by Carls [8] for abstract abelian varieties. In our case, we have explicit equations for the Richelot correspondences of curves, for which this theoretical machinery is not required and the convergence can be checked using classical criteria (valuation of the Jacobian matrix of the system of equations).

2.3 Reduction of the Moduli Subvariety

This section is based on the work of Goren [19] describing the reduction of an abelian surface.

Theorem 4 ([19]). *Let K be a cyclic quartic CM field and A an abelian variety having CM by \mathcal{O}_K the ring of integers of K . Let $\bar{\mathfrak{p}}$ be a prime of $\overline{\mathbb{Q}}$, $\mathfrak{p}_1 = \bar{\mathfrak{p}} \cap \mathcal{O}_K$ and $(p) = \mathfrak{p}_1 \cap \mathbb{Z}$. Assume that p is unramified in K . Then the reduction $A_{\bar{\mathfrak{p}}}$ of $A \pmod{\bar{\mathfrak{p}}}$ is determined by the decomposition of p in \mathcal{O}_K as follows:*

- (i) if $p = \mathfrak{P}_1\mathfrak{P}_2\mathfrak{P}_3\mathfrak{P}_4$ then $A_{\overline{\mathfrak{P}}}$ is ordinary and simple;
- (ii) if $p = \mathfrak{P}_1\mathfrak{P}_2$ then $A_{\overline{\mathfrak{P}}}$ is isomorphic to the product of two supersingular elliptic curves;
- (iii) if $p = \mathfrak{P}_1$ then $A_{\overline{\mathfrak{P}}}$ is isogenous but not isomorphic to a product of two supersingular elliptic curves.

For a non-normal quartic CM field, which is the generic case, an analogous theorem holds: depending on group theoretic considerations in the Galois group of the normal closure of K , one can decide whether the reduction of the Jacobian of a CM curve is ordinary, intermediate, or supersingular, and whether or not it is simple. We omit the details here and refer instead to Goren [19] for a precise statement.

These results are used at two places. First, they are required in the final curve construction step, to determine a prime of ordinary reduction, a necessary condition for cryptographic use. From the primes of ordinary reduction, we choose a prime p such that a solution to the Igusa class polynomials over \mathbb{F}_p gives a group order which is prime. Second, for the 2-adic method to work, the reduction modulo 2 must be ordinary, otherwise the canonical lift is not well-defined and the lifting algorithm does not apply. Given a CM field K , the theorem describes when there exists an ordinary curve defined over a finite field \mathbb{F}_{2^d} with CM by \mathcal{O}_K . As the input to our algorithm is an ordinary curve, rather than the CM field K , this theorem describes the condition at 2 on those CM fields which can be treated by our algorithm.

3 New Representation of the CM Variety

Before presenting our 2-adic CM method, we explain our modification to the representation of the ideal describing the CM invariants. In the classical CM method, Spallek [35] chose to compute three polynomials H_1, H_2 and H_3 , defined as

$$H_1 = \prod_{i=1}^{h_K^*} (X - j_1^{(i)}), H_2 = \prod_{i=1}^{h_K^*} (X - j_2^{(i)}) \text{ and } H_3 = \prod_{i=1}^{h_K^*} (X - j_3^{(i)}).$$

Subsequently Weng [40] formalized the classical CM method for genus two in terms of the same polynomials. However these polynomials determine an ideal $(H_1(j_1), H_2(j_2), H_3(j_3)) \subset \mathbb{Q}[j_1, j_2, j_3]$, of degree $h_K^*{}^3$, i.e. defining $h_K^*{}^3$ points $(j_1^{(i_1)}, j_2^{(i_2)}, j_3^{(i_3)})$, of which only the h_K^* solutions $(j_1^{(i)}, j_2^{(i)}, j_3^{(i)})$ determine valid CM curves.

In order to compute the equation of a CM curve, we need to test all $h_K^*{}^3$ candidate solutions to this system of equations to find one of the h_K^* which is known to have the correct endomorphism ring. For each solution we must apply Mestre’s algorithm [28] to find the corresponding curve, then to test a random point on the Jacobian to determine if the group of rational points has the correct order. This overhead is unnecessary since with a few additional relations among the (j_1, j_2, j_3) , we determine a complete set of relations for the CM invariants of the desired CM order.

The solution is to find some compact representation for the full ideal of class invariants. Beginning with the minimal polynomial of j_1 , $H_1(X) = \prod_{i=1}^{h_K^*} (X - j_1^{(i)}) \in \mathbb{Q}[X]$, we then use Lagrange interpolation to compute

$$G_k(X) = \sum_{i=1}^{h_K^*} j_k^{(i)} \prod_{\substack{\ell=1 \\ \ell \neq i}}^{h_K^*} \frac{X - j_1^{(\ell)}}{j_1^{(i)} - j_1^{(\ell)}} \in \mathbb{Q}[X], \text{ for } k = 2, 3.$$

This solves the problem of having an incomplete specification for the ideal of invariants, since $j_k = G_k(j_1)$ are uniquely determined by any root j_1 of $H_1(X)$. To determine a CM curve over \mathbb{F}_p , we solve for a root \bar{j}_1 of $H_1(X) \pmod p$ which determines $\bar{j}_2 = G_2(\bar{j}_1)$ and $\bar{j}_3 = G_3(\bar{j}_1)$, and use Mestre’s algorithm to determine a CM curve from the triple $(\bar{j}_1, \bar{j}_2, \bar{j}_3)$.

Modified Lagrange interpolation. The above construction provides an exact description of the CM invariants, but we observe empirically that the coefficient sizes of G_k , in comparison with those for H_k , are larger by a factor of three to four. However, in the formulae for G_k , we can pull out the factor $H_1'(j_1^{(i)})^{-1} = \prod_{k \neq i} (j_1^{(i)} - j_1^{(k)})^{-1}$. Therefore instead of using G_k we consider the polynomials

$$\widehat{H}_k(X) = \sum_{i=1}^{h_K^*} j_k^{(i)} \prod_{\substack{\ell=1 \\ \ell \neq i}}^{h_K^*} (X - j_1^{(\ell)}) \in \mathbb{Q}[X] \text{ for } k = 2, 3,$$

which recover the lost factor, and have coefficients of the same order of magnitude as H_k . The defining relations for our CM invariants can now be expressed as

$$H_1(j_1) = 0, \quad H_1'(j_1)j_2 = \widehat{H}_2(j_1), \quad H_1'(j_1)j_3 = \widehat{H}_3(j_1).$$

In order to explain the decrease in the size of the polynomial coefficients, we make some assumptions to deal with a notion of size for the j -invariants we are manipulating. Let L be a number field containing all Galois conjugates $j_k^{(i)}$ of the j -invariants. We assume that there exists a notion of a logarithmic height function $h : L \rightarrow \mathbb{R}_{>0}$, measuring the size of elements, which satisfies the properties: $h(ab) = h(a) + h(b)$, and $h(a+b) \leq \max(h(a), h(b))$, for general a and b . We extend h to a height function on $L[X]$ by: $h(\sum_{i=0}^n a_i X^i) = \sum_{i=0}^n h(a_n)$. We also assume that all the j -invariants are random elements of bounded height S . We can then estimate the relative heights of our polynomials H_k , G_k , and \widehat{H}_k . We evaluate the size of H_k to be

$$h(H_k) \leq \sum_{i=1}^{h_K^*} iS = \frac{h_K^*(h_K^* + 1)}{2}S,$$

since the coefficients of H_k are symmetric polynomials in the $j_k^{(i)}$. A similar calculation for G_k and \widehat{H}_k gives $h(G_k) \leq 2h_K^*(h_K^* - 1)S$, and $h(\widehat{H}_k) \leq h_K^*(h_K^* - 1)S$.

Under the assumption that the j -invariants behave as random elements, we expect equality to hold for each bound. This analysis, although heuristic, agrees with the empirical results of the algorithm.

Remark 2. We emphasize the fact that this new representation applies both to the classical CM construction and to our new p -adic method that we present in the next section.

4 The 2-Adic CM Method

In this section we describe our algorithm for computing the Igusa class polynomials $H_1, \widehat{H}_2, \widehat{H}_3$ corresponding to a CM order. In the classical approach one starts from a CM field and computes the Igusa class polynomials. In our approach, the input is a genus 2 curve defined over a small finite field \mathbb{F}_{2^d} , for some small d , and we reconstruct the class polynomials associated to its canonical lift. The input curves for this construction are defined over a tiny field of no cryptographic interest, but via their canonical lift we find their class invariants over \mathbb{Q} , which can then be reduced modulo p to produce curves of cryptographic application over some large prime field \mathbb{F}_p . We note that the class polynomials we find may determine a proper irreducible factor of the CM class invariants, in the case the invariants fall into distinct Galois orbits. However, for their application to cryptography this only aids in the rational reconstruction phase of our algorithm.

The algorithm proceeds as follows. Since d is small, one can easily compute all the data related to the input curve \mathcal{C} , in particular the endomorphism ring \mathcal{O} of its Jacobian, which we assume to be the maximal order of a CM field K . The canonical lift of \mathcal{C} is then computed to a high precision, so that we can get a good 2-adic approximation of its Igusa invariants. Theorem 1 gives a way to predict the degree h_K^* of the class polynomials. From this information, if the precision is sufficient, there is a unique possibility left for the polynomials $H_1, \widehat{H}_2, \widehat{H}_3$. These can be computed by running the LLL algorithm on a matrix built from powers of the invariants of the canonical lift. Algorithm 1 gives a summary of the algorithm, and in the next two subsections we discuss the details.

4.1 Computing the Canonical Lift

Canonical lifts were introduced in cryptography for the purpose of point counting by Satoh [32] for elliptic curves. After many improvements by several people, this ended up in a very fast method that runs in a time which is almost-linear in the required precision. A precise description and comparison of the various methods in the elliptic case can be found in [39] to which we refer for additional reading. Two genus 2 variants have been introduced by Mestre [27,26], based on the Richelot isogeny or on the Borchardt mean. The latter variant has been developed in detail by Lercier and Lubicz [23].

Algorithm 1 The 2-adic CM method

Input : An ordinary genus 2 curve \mathcal{C} defined over \mathbb{F}_{2^d} having CM by an order \mathcal{O} ;

Output : $(H_{1,\text{irr}}, \widehat{H}_{2,\text{irr}}, \widehat{H}_{3,\text{irr}})$ which determine an irreducible factor of the class invariants $(H_1, \widehat{H}_2, \widehat{H}_3)$ of \mathcal{O} .

- 1: Compute the j -invariants of \mathcal{C} and choose an arbitrary lift to \mathbb{Z}_{2^d} ;
- 2: Compute the canonical lifts $(j_1, j_2, j_3) \in (\mathbb{Z}_{2^d})^3$, i.e. the j -invariants of the canonical lift of \mathcal{C} ;
- 3: Determine the degree h_K^* of $(H_1, \widehat{H}_2, \widehat{H}_3)$;
- 4: Apply the LLL algorithm with input h_K^* and powers of (j_1, j_2, j_3) ;
- 5: Retrieve the result of LLL, that is the polynomials $H_{1,\text{irr}}, \widehat{H}_{2,\text{irr}}$ and $\widehat{H}_{3,\text{irr}}$ verifying

$$H_{1,\text{irr}}(j_1) = 0, \quad H'_{1,\text{irr}}(j_1) \cdot j_2 = \widehat{H}_{2,\text{irr}}(j_1) \quad \text{and} \quad H'_{1,\text{irr}}(j_1) \cdot j_3 = \widehat{H}_{3,\text{irr}}(j_1);$$

- 6: Return the triple $(H_{1,\text{irr}}, \widehat{H}_{2,\text{irr}}, \widehat{H}_{3,\text{irr}})$.
-

For the present work, we used the former approach, based on Richelot isogenies, together with the asymptotically fast lifting algorithm of Harley. Since this is not well described in the literature, we say a few words about it.

The main point is that Richelot isogeny as described in [5] gives relations between the defining equations of genus 2 curves whose Jacobian are $(2, 2)$ -isogenous. We take equations in the Rosenhain form: $y^2 = x(x-1)(x-\lambda_0)(x-\lambda_1)(x-\lambda_\infty)$. Putting $\Lambda = (\lambda_0, \lambda_1, \lambda_\infty)$, we can realize the relations coming from Richelot isogeny as a system of polynomial maps $\Phi = (\Phi_1, \Phi_2, \Phi_3)$ from $\mathbb{Q}_{2^d}^6 = \mathbb{Q}_{2^d}^3 \times \mathbb{Q}_{2^d}^3$ to $\mathbb{Q}_{2^d}^3$, such that two curves of Rosenhain invariants Λ and Λ' have Jacobians related by a $(2, 2)$ -isogeny if and only if $\Phi(\Lambda, \Lambda') = 0$. Hence, according to Theorem 3, the Rosenhain invariants Λ of the canonical lift of the curve \mathcal{C} we are interested in must verify $\Phi(\Lambda, \Lambda^\sigma) = 0$. Before giving the explicit formulae for Φ , we sketch how Harley’s algorithm can be adapted to the multivariate setting.

Assume we have an approximation $\Lambda_0 \in \mathbb{Q}_{2^d}^3$ to the Rosenhain invariants Λ of the canonical lift, correct to precision 2^k . Let $\Lambda_1 \in \mathbb{Z}_{2^d}^3$ be such that $\Lambda = \Lambda_0 + 2^k \Lambda_1$. Then Λ satisfies the equation $\Phi(\Lambda, \Lambda^\sigma) = 0$, which rewrites as

$$0 = \Phi(\Lambda_0 + 2^k \Lambda_1, \Lambda_0^\sigma + 2^k \Lambda_1^\sigma) = \Phi(\Lambda_0, \Lambda_0^\sigma) + 2^k d\Phi(\Lambda_0, \Lambda_0^\sigma) \begin{bmatrix} \Lambda_1 \\ \Lambda_1^\sigma \end{bmatrix} \pmod{2^{2k}},$$

from which Λ_1 can be deduced. Indeed, since $\Phi(\Lambda_0, \Lambda_0^\sigma) \equiv 0 \pmod{2^k}$, the equation in Λ_1 can be restated as $A\Lambda_1 + B = 0$, where A is a 3×3 matrix over \mathbb{Z}_{2^d} , and B and Λ_1 are vectors in $\mathbb{Z}_{2^d}^3$. Another level of recursive Newton-lifting is used for solving this so-called Artin-Schreier equation.

In this brief description, we have freely assumed that computing σ is a cheap operation, which is unfortunately not true if one takes an arbitrary defining polynomial $f(x)$ for the extension field $\mathbb{Q}_{2^d} = \mathbb{Q}_2[x]/(f(x))$. The trick is to choose the polynomial $f(x)$ such that f divides $x^{2^d} - x$, which in turn implies that $t^\sigma = t^2$, where t is the defining element of the extension field. The computation

of such an f is done, again, by a Newton lifting algorithm based on the equation $f(x^2) = f(x)f(-x)$, which is easily seen to be satisfied by the polynomial we are looking for. We refer to [39] for a more precise description.

Let us now describe the polynomial maps Φ given by the Richelot’s isogeny. For clarity, we give them in an implicit form that introduces new intermediate variables. Let λ_0, λ_1 and λ_∞ be the starting Rosenhain invariants. The images $\lambda_0^\sigma, \lambda_1^\sigma$ and λ_∞^σ of λ_0, λ_1 and λ_∞ by the second power Frobenius automorphism are given by the following formulae:

$$\lambda_0^\sigma = \frac{(u_1 - v_\infty)(w_0 - v_0)}{(u_1 - v_0)(w_0 - v_\infty)}, \lambda_1^\sigma = \frac{(u_1 - u_\infty)(w_1 - v_0)}{(u_1 - v_0)(w_1 - v_\infty)} \text{ and } \lambda_\infty^\sigma = \frac{(u_1 - v_\infty)(u_\infty - v_0)}{(u_1 - v_0)(u_\infty - v_\infty)},$$

where $(u_1, u_\infty), (v_0, v_\infty)$ and (w_0, w_1) are the respective roots of the polynomials

$$\begin{aligned} &U^2 - 2\lambda_\infty U + \lambda_\infty(1 + \lambda_1) - \lambda_1, \\ &V^2 - 2\lambda_\infty V + \lambda_0\lambda_\infty, \text{ and} \\ &(\lambda_0 - 1 - \lambda_1)W^2 + 2\lambda_1 W - \lambda_0\lambda_1. \end{aligned}$$

Remark 3. We need to pay attention to the valuations of our Rosenhain invariants. Assuming that we begin with $\lambda_0 \equiv 0 \pmod 4, \lambda_1 \equiv 1 \pmod 4$ and $\text{val}(\lambda_\infty) = -2$, we choose the labeling of the roots of our quadratic polynomials such that $v_0, w_0 \equiv 0 \pmod 2, u_1, w_1 \equiv 1 \pmod 2$, and $\text{val}(u_\infty), \text{val}(v_\infty) < 0$, from which $\lambda_0^\sigma \equiv 0 \pmod 4, \lambda_1^\sigma \equiv 1 \pmod 4$ and $\text{val}(\lambda_\infty^\sigma) = -2$ follows.

4.2 Recognizing Class Polynomials in $\mathbb{Q}[X]$

In this section we explain how we use the LLL algorithm to recover the minimal polynomials over \mathbb{Z} of the canonical lifted j -invariants. Let $A = \langle b_1, \dots, b_m \rangle$ be a lattice and let $\det(A)$ be its determinant. Minkowski’s inequality gives the upper bound $\sqrt{m/2\pi e} \det(A)^{1/m}$, for the norm of the shortest lattice vector, and in a random lattice, one expects a minimal length vector to be close to this norm. The LLL algorithm outputs a basis of short vectors, and if we construct A to have a known vector $v \in A$ of norm much smaller than this bound, then, heuristically, it will be the shortest vector in A .

Let \mathbb{Z}_{2^d} be an extension of \mathbb{Z}_2 of degree d with \mathbb{Z}_2 -basis $1, w_1, \dots, w_{d-1}$. Let $\alpha \in \mathbb{Z}_{2^d}$ generate \mathbb{Z}_{2^d} , and $\tilde{\alpha}$ be an approximation of α modulo a high power of 2, say $\alpha \equiv \tilde{\alpha} \pmod{2^N}$. We assume that we know the degree s of its minimal polynomial $f(x) \in \mathbb{Z}[x]$, i.e. $f(x) = a_s x^s + \dots + a_0$ where the $(a_i) \subseteq \mathbb{Z}$ are unknown. The degree s of the minimal polynomial is the degree of an irreducible factor of Igusa class polynomials, whose degree is h_K^* . In order to determine the (a_i) , we determine a basis of the left kernel in \mathbb{Z}^{s+d+1} of the matrix

$$\begin{pmatrix} A \\ 2^N I_d \end{pmatrix}, \text{ where } A \text{ is the } (s+1) \times d \text{ matrix: } \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,(d-1)} \\ \vdots & & & \vdots \\ \alpha_{s,0} & \alpha_{s,1} & \cdots & \alpha_{s,(d-1)} \end{pmatrix},$$

with $\alpha_{j,k}$ defined by $\alpha^j = \alpha_{j,0} + \alpha_{j,1}w_1 + \dots + \alpha_{j,(d-1)}w_{d-1}$.

In order to compute the basis of the left kernel, we apply the LLL algorithm in the same way as described in [10]. This kernel is a lattice Λ , in which the coefficients of the minimal polynomial of α are part of a short vector. Indeed, if a_0, \dots, a_s are integers with $|a_i| \ll 2^N$ such that $a_s \alpha^s + \dots + a_0 \equiv 0 \pmod{2^N}$, then $(a_0, \dots, a_s, \varepsilon_1, \dots, \varepsilon_d)$ will be a short vector in Λ , for appropriate integers (ε_i) . Any other solution that is not proportional to the (a_i) will differ by an element of $\Lambda_0 + 2^N \mathbb{Z}^{s+d+1}$, where Λ_0 is generated by the $c_d \alpha^{d+i} + \dots + c_0 \alpha^i \equiv 0 \pmod{2^N}$, $1 \leq i \leq s - d$, coming from the minimal polynomial $g(x) = c_d x^d + \dots + c_0$ of α in $\mathbb{Z}_2[x]$ having arbitrary coefficients in \mathbb{Z}_2 . If the precision N is sufficiently high, we expect the unique solution (a_0, \dots, a_s) to appear as the shortest vector in the LLL-reduced lattice basis.

We remark that we can easily compute the image of (j_1, j_2, j_3) by the Frobenius σ and therefore we have access to the powers of (j_1, j_2, j_3) and $(j_1^{\sigma^i}, j_2^{\sigma^i}, j_3^{\sigma^i})$ for $i \in [1, d]$. Therefore we can use this information as input of our LLL algorithm. It implies a more complicated recognition phase where we have to use the subresultant algorithm to recognize our minimal polynomials. Moreover an explosion of the coefficient size in the course of the algorithm leads us to use modular arithmetic and the Chinese remainder theorem for our computations.

5 Complexity and Comparison with Other Methods

5.1 Complexity of the 2-Adic CM Method

The two costly steps of the 2-adic CM method are the computation of the canonical lift and the reconstruction of the polynomials using LLL. Those two steps highly depend on the precision k at which we have to compute the canonical lift in order to recover the full polynomials. This precision k depends itself on the sizes of the polynomial $H_1, \tilde{H}_2, \tilde{H}_3$, for which no bound (that would depend on the class number of K) is known. Hence we shall keep k in our formulae, although this is not a parameter under control.

By using advanced algorithms coming from point counting, the canonical lift computation takes a time which is essentially linear in the precision k . More precisely it has a complexity $O(M(dk) \log(k))$ where $M(dk)$ is the time for multiplying integers with dk bits, that is $O(dk)$ up to logarithmic factors.

The complexity of the LLL step involves the further parameter h_K^* , which is the degree of the polynomials we are trying to reconstruct. Using the classical LLL algorithm, we end up with a complexity of $O((h_K^* + d)^6 k^3)$. The L^2 variant of Nguyễn and Stehlé [30] has a better general complexity of $O((h_K^* + d)^5 (h_K^* + d + k)k)$, and in our case the structure of the lattice gives us an improved complexity of $O((h_K^* + d)^4 (h_K^* + d + k)k)$.

Now we will analyze what we could expect from the PSLQ algorithm. In [1], given an input of $h_K^* + d$ complex numbers whose integer relation is bounded by 2^k , the PSLQ algorithm is claimed to have a number of iterations in $O((h_K^* + d)^3 + (h_K^* + d)^2 k)$. Each iteration consists of four steps. Both for the complexity in the dimension and in the precision the bottleneck step is the third step, Hermite's

reduction and matrix multiplication. Therefore the complexity of one iteration is $O((h_K^* + d)^3 k)$. The total complexity of PSLQ seems to be $O((h_K^* + d)^6 k + (h_K^* + d)k^2)$ thus we do not expect any improvement from using a 2-adic version of PSLQ.

5.2 Comparison with Other Methods

The comparison with the classical CM method [35,40] is only valid for inputs at which their outputs coincide, since the inputs to each algorithm is different. In the 2-adic method one treats only CM fields where the ideal (2) has a special structure, and moreover the input is not the field but a hyperelliptic curve over a small finite field. In the classical CM method one starts directly from a CM field, with the requirement that the class number of the real subfield is 1. The main advantage of the 2-adic method compared to the classical method is that the complex floating point evaluation of theta constants at the period matrices (which is the bottleneck in the classical method) is replaced by a p -adic canonical lifting procedure for which we have precise control over precision and precision loss (there is none). Furthermore, the time-complexity of the evaluation of theta constants is quadratic in the required precision, whereas the canonical lift is essentially linear in the precision. On the other hand, the drawback of the 2-adic CM method is that the reconstruction step is much more expensive than in the classical case, since the step of building a polynomial from its roots is replaced by a call to the LLL algorithm. In this later case, the complexity becomes again quadratic in the precision. In other words, by changing the method, we have moved the bottleneck of the approach from the first step to the second step.

We can also compare to the CRT approach [9,16]. In that case, to be able to build a class polynomial whose coefficients have k bits, one needs to use $O(k)$ small finite fields \mathbb{F}_{p_i} , where p_i is $O(k)$. Finding the appropriate curves implies $O(p_i^3)$ steps for each p_i , since we essentially have to enumerate all isomorphism classes over the field \mathbb{F}_{p_i} . Hence the complexity is more than quadratic in the precision, so that the CRT method is not competitive with the other methods in terms of required precision. This ignores the endomorphism ring computation which is exponential in p_i in the worst case (but might be controlled by a more selective sieving for CRT primes).

5.3 Experiments

All of the experiments we carried out were written using `Magma` [12] and `C` routines. The 2-adic arithmetic is taken from an experimental `gmp`-style library called `Mploc` which was developed by E. Thomé [37]. It currently contains far more than the 2-adic arithmetic, including efficient arithmetic in \mathbb{Q}_p , $\mathbb{Q}_p[X]$, and extensions of \mathbb{Q}_p . We use `NTL` [34] library for the floating-point LLL routine, as at the time we developed our program, Stehlé's LLL `C` routines were not available [36]. All the experiments were conducted on a 2.4 GHz Athlon 64. On such a computer, computing irreducible factors of Igusa class polynomials of degree less than twenty is a question of minutes.

Example. Let \mathcal{C} be the curve of equation $y^2 + h(x)y + f(x) = 0$ over $\mathbb{F}_{32} = \mathbb{F}_2[t]/(t^5 + t^2 + 1)$, with $f(x) = x^5 + t^{20}x^3 + t^{17}x^2 + t^{19}x$ and $h(x) = x^2 + t^9x$. The curve is ordinary and has CM by the maximal order of $K = \mathbb{Q}(i\sqrt{75} + 12\sqrt{17})$. The field K is non-normal and its class number is 50; so we have $h_K^* = 100$ isomorphism classes of principally polarized abelian varieties.

Looking for a minimal polynomial of the lifted value of j_1 , the LLL algorithm produced a plausible answer of degree 50. A more subtle analysis of the Galois theory in fact predicts that the class polynomial of degree 100 is reducible over the rationals, splitting in two factors of degree 50. Using our method, we produce one of these two factors $H_1(X)$, with the corresponding polynomials $\hat{H}_2(X)$ and $\hat{H}_3(X)$. The leading coefficient of H_1 is $3^{50}11^{156}17^{60}23^{72}41^{24}73^{12}83^{12}181^{48}691^{12}$, consistent with the theory of Goren-Lauter [20], and reduction at a large prime gave rise to a Jacobian whose group of rational points agreed with the expected order for this CM field.

For this example, we used a 2-adic precision of 65000 bits, and the running time to lift the curve and compute the invariants was 20 seconds. The subsequent lattice reductions took about one day. This confirms that the bottleneck is in the second step, as predicted by the complexity estimates, and suggests that an improved strategy would be to lift additional j -invariants to reduce the size of the lattice in the reduction phase.

6 Conclusion and Perspectives

This work presents a new p -adic method for building Igusa class polynomials for genus two curves, that can be used to efficiently produce CM curves suitable for cryptography. Our method makes use of p -adic lifting techniques borrowed from point counting algorithms. The algorithm performs well in practice and has allowed us to treat much larger class numbers than previously reported in the literature.

In order to deal with such large degree class polynomials, we were led to introduce a new representation for the ideal of CM points, so that the final step of the CM method — namely reducing the polynomials modulo an appropriate prime p and constructing the corresponding curve equation — no longer requires a combinatorial search for one valid tuple of invariants for each h_K^* tuple when using class polynomials of degree h_K^* .

Our work is based on curves of characteristic 2, which places a restriction on which CM fields we can treat. This is analogous to the condition on discriminants treatable by the CM construction in genus 1 using reduced class polynomials in terms of Weber functions. Extending this algorithm to other small characteristics p would impose an independent condition so that more CM fields could be treated. Such algorithms are the subject of ongoing investigation, motivated by this research.

As the discussion of complexity issues indicates, the different methods for building Igusa class polynomials (complex analytic, p -adic analytic, CRT) all have advantages and limitations. Combining them in order to take advantage of

the best of each method is something that should be explored. For example, an algebraic formula for the exact leading coefficient of the Igusa class polynomials (see [20]) would have benefit to a greater or lesser extent in each of these methods. We note that the bottleneck of the classical CM method is the evaluation of theta constants. Recently, Dupont [15] developed new algorithms for this task, yielding a huge performance improvement for the classical CM method. Further investigation of the limiting steps for the classical and p -adic methods will determine in the end which algorithm applies most effectively to a given problem.

References

1. S. Arno, D. H. Bailey, and H. R. P. Ferguson. Analysis of PSLQ, an integer relation finding algorithm. *Math. Comp.*, 68(225):351–369, January 1999.
2. R. Avanzi. Aspects of hyperelliptic curves over large prime fields in software implementations, 2003. Preprint (available at <http://eprint.iacr.org/2003/253>).
3. A. Borel, S. Chowla, C. S. Herz, K. Iwasawa, and J.-P. Serre. *Seminar on complex multiplication*. Number 21 in Lecture Notes in Math. Springer, 1966.
4. Z. I. Borevitch and I. R. Shafarevich. *Number theory*, volume 20 of *Pure and Applied Mathematics*. Academic Press Inc., New-York, 1966.
5. J.-B. Bost and J.-F. Mestre. Moyenne arithmético-géométrique et périodes de courbes de genre 1 et 2. *Gaz. Math. Soc. France*, 38:36–64, 1988.
6. R. Bröker and P. Stevenhagen. Elliptic curves with a given number of points. In D. Buell, editor, *ANTS-VI*, vol. 3076 of LNCS, pages 117–131. Springer-Verlag, 2004.
7. R. M. Bröker. *Constructing elliptic curves of prescribed order*. PhD thesis, Thomas Stieltjes Institute for Mathematics, 2006.
8. R. Carls. *A generalized arithmetic geometric mean*. PhD thesis, Rijksuniversiteit Groningen, 2004.
9. J. Chao, K. Matsuo, H. Kawashiro, and S. Tsujii. Construction of hyperelliptic curves with CM and its application to cryptosystems. In T. Okamoto, editor, *ASIACRYPT 2000*, vol. 1976 of LNCS, pages 259–273. Springer-Verlag, 2000.
10. H. Cohen. *A course in algorithmic algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993. Second corrected printing, 1995.
11. H. Cohen and H. W. Lenstra, Jr. Heuristics on class groups of number fields. In *Number theory, Noordwijkerhout 1983*, volume 1068 of *Lecture Notes in Math.*, pages 33–62. Springer, Berlin, 1984.
12. The University of Sydney Computational Algebra Group. Magma online handbook, 2006. <http://magma.maths.usyd.edu.au/magma/htmlhelp/MAGMA.htm>.
13. J.-M. Couveignes and T. Henocq. Action of modular correspondences around CM points. In C. Fieker and D. R. Kohel, editors, *ANTS-V*, vol. 2369 of LNCS, pages 234–243. Springer-Verlag, 2002.
14. M. Deuring. Die Typen der Multiplikatorringe elliptischer Funktionenkörper. *Abh. Math. Sem. Hansischen*, 14:197–272, 1941.
15. R. Dupont. *Moyenne arithmético-géométrique, suites de Borchartd et applications*. PhD thesis, École polytechnique, 2006.
16. K. Eisentrager and K. Lauter. Computing Igusa class polynomials via Chinese Remainder Theorem. 2004. Preprint (available at <http://arxiv.org/abs/math.NT/0405305>), 2004.

17. P. Gaudry. Fast genus 2 arithmetic based on Theta functions, 2005. Preprint (available at <http://eprint.iacr.org/2005/314>).
18. P. Gaudry and É. Schost. Construction of secure random curves of genus 2 over prime fields. In C. Cachin and J. Camenisch, editors, *Eurocrypt 2004*, vol. 3027 of LNCS, pages 239–256. Springer–Verlag, 2004.
19. E. Z. Goren. On certain reduction problems concerning abelian surfaces. *Manuscripta math.*, 94:33–43, 1997.
20. E. Z. Goren and K. Lauter. Class invariants for quartic CM fields. Preprint (available at <http://arxiv.org/abs/math.NT/0404378>), 2004.
21. N. Koblitz. *p-adic Numbers, p-adic Analysis and Zeta-Functions*, volume 58 of *Graduate Texts in Mathematics*. Springer–Verlag, 1984.
22. T. Lange and M. Stevens. Efficient doubling on genus two curves over binary fields. vol. 3357 of LNCS, pages 170–181. Springer–Verlag, 2005. In H. Handschuh and M.A. Hasan, editors, *SAC 2004*.
23. R. Lercier and D. Lubicz. A quasi quadratic time algorithm for hyperelliptic curve point counting. To appear in *J. Ramanujan Math. Soc.*
24. R. Lercier and E. Riboulet-Deyris. Elliptic curves with complex multiplication. Posting to the Number Theory List, available at <http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0401&L=nmbrthry&P=R305>, 2004.
25. J. Lubin, J.-P. Serre, and J. Tate. Elliptic curves and formal groups. In *Lecture notes prepared in connection with the seminars held at the Summer Institute on Algebraic Geometry, Whitney Estate, Woods Hole, Massachusetts, July 6-July 31, 1964*, 1964. Scanned copies available at <http://www.ma.utexas.edu/users/voloch/1st.html>.
26. J.-F. Mestre. Algorithmes pour compter des points de courbes en petite caractéristique et en petit genre. Talk given in Rennes in March 2002. Notes written by D. Lubicz.
27. J.-F. Mestre. Utilisation de l’AGM pour le calcul de $E(F_{2^n})$. Lettre adressée à Gaudry et Harley, Décembre 2000.
28. J.-F. Mestre. Construction de courbes de genre 2 à partir de leurs modules. In T. Mora and C. Traverso, editors, *Effective methods in algebraic geometry*, vol. 94 of *Progr. Math.*, pages 313–334. Birkhäuser, 1991.
29. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5), May 2001.
30. P. Nguyễn and D. Stehlé. Floating-point LLL revisited. In *Eurocrypt 2005*, vol. 3494 of LNCS, pages 215–233. Springer–Verlag, 2005.
31. J. Pelzl, T. Wollinger, J. Guajardo, and C. Paar. Hyperelliptic curve cryptosystems: Closing the performance gap to elliptic curves. Preprint, 2003.
32. Takakazu Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.*, 15(4):247–270, 2000.
33. G. Shimura. *Abelian Varieties with complex multiplication and modular functions*. Princeton University Press, revised edition, 1998.
34. V. Shoup. NTL: A library for doing number theory. <http://www.shoup.net/ntl/>.
35. A.-M. Spallek. *Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen*. PhD thesis, Universität Gesamthochschule Essen, July 1994.
36. D. Stehlé. `fp111-1.2` a lattice LLL-reduction program, 2006. available at <http://www.loria.fr/~stehle>.
37. E. Thomé. Multi-Precision for LOfal-fields library, 2006. still under development, see <http://www.loria.fr/~thome>.
38. P. van Wamelen. Examples of genus two CM curves defined over the rationals. *Math. Comp.*, 68(225):307–320, January 1999.

- 39. F. Vercauteren. *Computing zeta functions of curves over finite fields*. PhD thesis, Katholieke Universiteit Leuven, 2003.
- 40. A. Weng. *Konstruktion kryptographisch geeigneter Kurven mit komplexer Multiplikation*. PhD thesis, Universität GH Essen, 2001.

A Cryptographic CM Curve Generation on One Example

We start with the curve \mathcal{C} of equation $y^2 + h(x)y + f(x) = 0$ over $\mathbb{F}_8 = \mathbb{F}_2[t]/(t^3 + t + 1)$, with $f(x) = x^5 + t^6x^3 + t^5x^2 + t^3x$ and $h(x) = x^2 + x$. The curve is ordinary and has complex multiplication by the maximal order of $K = \mathbb{Q}(i\sqrt{23 + 4\sqrt{5}})$. The field K is non-normal and its class number is 3; so we have 6 isomorphism classes of principally polarized abelian varieties. We apply our algorithm and compute the canonical lift of \mathcal{C} to high precision (in fact, a posteriori, we see that 1200 bits are enough) and get its invariants. From this we reconstruct the minimal polynomial H_1 and the corresponding \widehat{H}_2 and \widehat{H}_3 . As expected, the degree of H_1 is 6.

$$\begin{aligned}
 H_1 &= 2^{18}5^{36}7^{24} T^6 \\
 &- 11187730399273689774009740470140169672902905436515808105468750000 T^5 \\
 &+ 501512527690591679504420832767471421512684501403834547644662988263671875000 T^4 \\
 &- 10112409242787391786676284633730575047614543135572025667468221432704263857808262923 T^3 \\
 &+ 118287000250588667564540744739406154398135978447792771928535541240797386992091828213521875 T^2 \\
 &- 2^4 3^{50} 5^{10} 11^4 13^4 53^4 701^4 16319^4 69938793494948953569198870004032131926868578084899317 T \\
 &+ 3^{60} 5^{15} 23^5 409^5 179364113^5 \\
 \widehat{H}_2 &= 2^{-3} (2734249284974589542086559782016563911333032280921936035156250000 T^5 \\
 &+ 57554607277149797568849387967258354564256002479144001401149377453125000000 T^4 \\
 &+ 2402137816085408582966361480412923409977297040376760501014543382338189483861887923 T^3 \\
 &- 75691166837057576824962404339816428897154828109931810138346946500235981947587900092046875 T^2 \\
 &+ 2^4 3^{48} 5^{10} 3582851967081231217443096939126403484719666514876459782054400437 T \\
 &- 3^{58} 5^{15} 11^4 13^2 23^3 409^3 23879^4 179364113^3 370974539856105277) \\
 \widehat{H}_3 &= 2^{-4} (200620022972650193875396249949338812342692117691040003906250000 T^5 \\
 &- 23006467431764975697282545882188900514908468992554759536043135578125000000 T^4 \\
 &+ 615017294619678068611319414718144161545088218260214211563850151291136646894987547 T^3 \\
 &- 14310698742415340178789612716269299249317950024503557714370659520249839645781463819312875 T^2 \\
 &- 2^4 3^{46} 5^8 13^4 61^4 18373951326869^4 25713288587261208212107985724468058651509734160907 T \\
 &+ 3^{55} 5^{13} 23^2 409^2 23561^4 440131^4 179364113^2 451986402352017881724712641689)
 \end{aligned}$$

From the Newton polygon of H_1 for the 2-adic valuation, we see that there are three roots that have valuation 0, and the others have negative valuation. Hence only three of the curves have good reduction modulo 2. However, since H_1 is irreducible over \mathbb{Q} , the 2-adic lifted invariants of any of the three conjugate curves yields the whole H_1 .

Choosing the 120-bit prime $p = 954090659715830612807582649452910809$, and solving a norm equation in the endomorphism ring \mathcal{O}_K , we know that a solution (j_1, j_2, j_3) to the Igusa class polynomials gives the invariants of a genus 2 curve whose Jacobian has prime order

$$910288986956988885753118558284481029311411128276048027584310525408884449$$

of 240-bits. We find a corresponding curve:

$$\begin{aligned}
 \mathcal{C} : y^2 &= x^6 + 827864728926129278937584622188769650 x^4 \\
 &+ 102877610579816483342116736180407060 x^3 \\
 &+ 335099510136640078379392471445640199 x^2 \\
 &+ 351831044709132324687022261714141411 x \\
 &+ 274535330436225557527308493450553085
 \end{aligned}$$

and a test of a random point on the Jacobian verifies the group order.

Extending Scalar Multiplication Using Double Bases

Roberto Avanzi^{1,*}, Vassil Dimitrov²,
Christophe Doche³, and Francesco Sica^{4,**}

¹ Faculty of Mathematics and Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany
roberto.avanzi@ruhr-uni-bochum.de

² Advanced Technology Information Processing Systems laboratory,
Centre for Informations Security and Cryptography, University of Calgary, Canada
dimitrov@atips.ca

³ Department of Computing
Macquarie University, North Ryde, NSW 2109, Australia
doche@ics.mq.edu.au

⁴ Department of Mathematics and Computer Science – Acecrypt
Mount Allison University, Sackville, Canada
fsica@mta.ca – <http://www.acecrypt.com>

Abstract. It has been recently acknowledged [4,6,9] that the use of double bases representations of scalars n , that is an expression of the form $n = \sum_{e,s,t} (-1)^e A^s B^t$ can speed up significantly scalar multiplication on those elliptic curves where multiplication by one base (say B) is fast. This is the case in particular of Koblitz curves and supersingular curves, where scalar multiplication can now be achieved in $o(\log n)$ curve additions.

Previous literature dealt basically with supersingular curves (in characteristic 3, although the methods can be easily extended to arbitrary characteristic), where $A, B \in \mathbb{N}$. Only [4] attempted to provide a similar method for Koblitz curves, where at least one base must be non-real, although their method does not seem practical for cryptographic sizes (it is only asymptotic), since the constants involved are too large.

We provide here a unifying theory by proposing an alternate recoding algorithm which works in all cases with *optimal* constants. Furthermore, it can also solve the until now untreatable case where both A and B are non-real. The resulting scalar multiplication method is then compared to standard methods for Koblitz curves. It runs in less than $\log n / \log \log n$ elliptic curve additions, and is faster than any given method with similar storage requirements already on the curve K-163, with larger improvements as the size of the curve increases, surpassing 50% with respect to the τ -NAF for the curves K-409 and K-571. With respect of windowed methods, that can approach our speed but require $O(\log(n) / \log \log(n))$ precomputations for optimal parameters, we offer the advantage of a fixed, small memory footprint, as we need storage for at most two additional points.

* Partially supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

** This work was partially supported by a NSERC Discovery Grant.

1 Introduction

In cryptographic algorithms designed around elliptic curves, the most expensive part is the scalar multiplication nP , where P lies on the curve. In order to speed up this computation, it was proposed already at a very early stage of their use to adopt special families of curves where a large multiple of P can be computed very quickly. This is the case of endomorphism curves [15] or Koblitz curves E_a [17].

We will examine more closely this latter class of curves. Defined over \mathbb{F}_{2^p} , they are endowed with the *Frobenius endomorphism* τ of the rational point group E_a (\mathbb{F}_{2^p}). Now, τP is a large multiple of P which can be computed in time $O(1)$ using normal bases or $O(\mathbf{p})$ using polynomial bases. The map τ is also identified with a complex root of an equation of the form $\tau^2 \pm \tau + 2 = 0$ that depends only on the curve equation. Using τ , one can devise good scalar multiplication algorithms, see §§ 2.3, 2.5 and 2.6. All these algorithms compute nP with¹ $\Omega(\log n)$ costly curve operations (such as a doubling or an addition). We call these algorithms *linear* (in the number of curve operations with respect to the bit size of the field), since also the number of curve operations is $O(\log n)$. There are two ways of improving over these algorithms: either we devise algorithms with lower complexity (sublinear methods), or we reduce the number of group operations by some multiplicative factor. *We deal here with the former paradigm.*

The novelty of our approach is to combine the use of τ with double bases, first introduced in elliptic curve cryptography in [11]. To achieve this, we consider a more general setting of double base number systems (DBNS) that can be applied also to other classes of curves, such as supersingular curves over fields of characteristic 3, where in place of the Frobenius the fast operation is point tripling. We show how to find decompositions

$$n = \sum_{i=0}^{k-1} (-1)^{e_i} A^{s_i} B^{t_i}$$

with (A, B) a suitable pair of algebraic integers (such as $(2, 3)$, $(3, \tau)$, or $(\bar{\tau}, \tau)$) s_i, t_i nonnegative integers and $e_i \in \{0, 1\}$. The length k of this expansion is $O(\log n / \log \log n)$. We reveal, similarly to [6], a scalar multiplication algorithm with cost $O(\log n / \log \log n)$ curve operations in presence of a fast group endomorphism. We call such an algorithm *sublinear*, when the number of curve operations over the bit size of the field goes to zero.

This is a first instance of a practical sublinear scalar multiplication algorithm with very little precomputations (which depend only on \mathbf{p} , not the curve or the point P) or storage requirements ($O(\log \mathbf{p})$ bits). We provide some computational comparisons with other methods to show that even on 163-bit curves, our method yields better results.

¹ We use the notation $\Omega(x)$ to mean $> cx$ for some positive c .

2 Background Material

2.1 Double Bases

Following [8], albeit with a slightly different notation, we will call a (A, B) -integer a number which can be written as $A^i B^j$ for some nonnegative integers i, j . We will extend the definition to algebraic integers, more precisely, integers in $\mathbb{Z}[\tau]$. We will also allow $A, B \in \mathbb{Z}[\tau]$. We define a (A, B) -integer expansion of n as a decomposition of n into a sum of (possibly signed) (A, B) -integers. Sometimes this will be also called a DBNS (A, B) recoding.

2.2 Koblitz Curves

For a general presentation of Koblitz curves, we refer to [13, § 15.1.1]. A Koblitz curve E_a is an elliptic curve defined over \mathbb{F}_{2^p} , with equation

$$E_a \quad : \quad y^2 + xy = x^3 + ax^2 + 1 \quad . \quad (1)$$

Here $a = 0$ or 1 , and \mathbf{p} is a prime chosen so to make the order of the group of points $E_a(\mathbb{F}_{2^p})$ equal to twice if $a = 1$ (resp. four times if $a = 0$) a prime number, for at least one choice of a . A point $P \in E_a(\mathbb{F}_{2^p})$ is then randomly chosen with order equal to that large prime. In view of Hasse’s theorem, which states that $|\#E_a(\mathbb{F}_{2^p}) - 2^p - 1| < 2^{\frac{p}{2}+1}$, this means that we can choose P so that $\text{ord } P$ is very close to 2^{p-1} if $a = 1$ and to 2^{p-2} if $a = 0$. Since E_a has coefficients in \mathbb{F}_2 , the Frobenius map $\tau(x, y) = (x^2, y^2)$ is an endomorphism of $E_a(\mathbb{F}_{2^p})$. Since squaring is a linear operation in characteristic two, computing τP is also linear and takes time $O(\mathbf{p})$. If normal bases are used to represent elements of \mathbb{F}_{2^p} , then computing τP is much faster, since it amounts to making two rotations, which is essentially free.

We can view τ as a complex number of norm 2 satisfying the quadratic equation $\tau^2 - (-1)^{1-a}\tau + 2 = 0$, since for any P on the curve, $\tau^2 P + 2P = (-1)^{1-a}\tau P$. Explicitly, $\tau = \frac{(-1)^{1-a} + \sqrt{-7}}{2}$. We will also make use of the conjugate $\bar{\tau} = (-1)^{1-a} - \tau$ of τ . This corresponds to the dual of the Frobenius endomorphism.

2.3 The τ -NAF for Koblitz Curves

All facts here are stated without proofs: These are found in [24,25].

Let us consider the Koblitz curve E_a defined over \mathbb{F}_{2^p} by equation (1), with base point P , and let τ denote the Frobenius endomorphism. We have seen that we can view $\tau(P)$ as *multiplication by τ* and let $\mathbb{Z}[\tau]$ operate on P , but in fact there exists an integer λ such that $\tau(P) = \lambda P$, and thus τ operates on the whole subgroup generated by P like multiplication by λ .

The τ -adic non-adjacent form (τ -NAF for short) of an integer $z \in \mathbb{Z}[\tau]$ is a decomposition $z = \sum_i z_i \tau^i$ where $z_i \in \{0, \pm 1\}$ with the *non-adjacency* property $z_j z_{j+1} = 0$, similarly to the classical NAF [21]. The average *density* (that is the average ratio of non-zero bits related to the total number of bits) of a τ -NAF is $1/3$. Each integer z admits a unique τ -NAF.

The length of the τ -NAF expansion of a randomly chosen scalar n is $\approx 2\mathbf{p}$, whereas the bit length of n is $\approx \mathbf{p}$. But, for any point $P \in E_a(\mathbb{F}_{2^{\mathbf{p}}}) \setminus E_a(\mathbb{F}_2)$, $\tau^{\mathbf{p}}P = P$ and $\tau P \neq P$.

Since the ring $\mathbb{Z}[\tau]$ is Euclidean we can take the remainder ζ of $n \bmod \frac{\tau^{\mathbf{p}}-1}{\tau-1}$ and use it in place of n . This ζ will have smaller norm than that of $(\tau^{\mathbf{p}}-1)/(\tau-1)$, and thus length at most \mathbf{p} . Its τ -NAF is called the *reduced* τ -NAF of n and when P has prime order, it can be shown that $nP = \zeta P$.

The double-and-add scalar multiplication algorithm is a Horner scheme for the evaluation of nP using the binary expansion of $n = \sum_{i=0}^{\ell} n_i 2^i$ as $\sum_{i=0}^{\ell} n_i 2^i P$. In a similar way we can evaluate $zP = \sum_i z_i \tau^i(P)$ by a Horner scheme, and the corresponding algorithm is called a τ -and-add algorithm. It is much faster than the double-and-add scheme on Koblitz curves because Frobenius evaluations are much faster than doublings.

2.4 Point Halving

Point halving (see [16] and [22,23]) is a technique to improve the performance of cryptosystems based on binary elliptic curves. The idea is to replace, in the double-and-add algorithm for scalar multiplication, doublings $2Q$ by halvings $\frac{1}{2}Q = \frac{\text{ord } Q + 1}{2}Q$. Even though halving is not as fast as a Frobenius operation, it is much faster than doubling (between two and three times faster), according to literature [16,22,23] as well as [14].

2.5 Inserting a Halving in the τ -Adic Scalar Multiplication

In [1] a single point halving is inserted in the “ τ -and-add” scalar multiplication. This brings a non-negligible speedup (up to 14%) with respect to the use of the τ -NAF, but is not optimal. In [3] the method is refined in order to bring the speedup to 25%, and the resulting method is proved optimal among similar methods that do not require any precomputation. The basic idea in both approaches is to express nP as $\sum_i e_{0,i} \tau^i(P) + \sum_i e_{1,i} \tau^i(Q)$ with $Q = \frac{1}{2}P$ and a smaller total Hamming weight of the $e_{j,i}$ ’s. The τ -and-add loop is repeated two times: first $\sum_i e_{1,i} \tau^i(P)$ is computed, then the result is halved and a second τ -and-add loop is performed like for the computation of $\sum_i e_{0,i} \tau^i(P)$, but starting with the result just obtained in place of 0.

2.6 Further Developments in τ -Adic Representations

The authors of [19] generalize the approach of [1] to expressions of the form $\sum_i e_{0,i} \tau^i(P) + \sum_i e_{1,i} \tau^i(f_1(P)) + \dots + \sum_i e_{2^u-2-1,i} \tau^i(f_{2^u-2-1}(P))$, where 1 and the f_j are representants of the residue classes modulo τ^u in the ring $\mathbb{Z}[\tau]$ which are coprime to τ , and $e_{j,i} \in \{0, \pm 1\}$. Such an expression can be obtained from a τ -adic windowed recoding [25]. If a window of width u is used, then the τ -and-add loop is performed 2^{u-2} times in place of two times as in the method of § 2.5. Thus, the number of Frobenius operations can increase exponentially with u . To ensure that this does not become a performance problem if polynomial

bases are used, a technique from [20] is adopted to convert between normal and polynomial bases as required to quickly compute iterated Frobenius operations.

At the end of the τ -and-add loop corresponding to the digit f_j the partial result must be multiplied by f_{j+1}/f_j before starting the τ -and-add loop corresponding to the next digit f_{j+1} . The relations between the f_j 's and their inverses must then be given explicitly. In [19] this is done for $w = 5$. Even though the authors cannot present the results in a completely general way, in the case described in [19] the reduction in memory consumption (or, equivalently, the speed-up with respect to other methods with no precomputations) is noteworthy. In order to generalize their approach the digit set itself has to be modified. In [2] it is shown how to do so.

2.7 Supersingular Elliptic Curves in Characteristic 3

We refer to [18] for generalities on supersingular elliptic curves. We will consider the curves E_b defined over \mathbb{F}_{3^m} by the Weierstraß equations [5]

$$y^2 = x^3 - x + b$$

with $b = \pm 1$. On these curves, the tripling operation sends $P = (x, y)$ to $3P = (x^9 - b, -y^9)$, meaning that point tripling is essentially equivalent to two Frobenius and its cost will be considered negligible.

3 Theoretical Preliminaries

All the new results proving the sublinearity of the new DBNS decompositions are based on the following propositions. These results appears naturally in any elementary number theory book during the proof of the structure theorem for $(\mathbb{Z}/m)^*$, the multiplicative group of invertible classes modulo m . In the sequel, we let \mathcal{R} be a unique factorization domain containing \mathbb{Z} (we will consider in practice $\mathcal{R} = \mathbb{Z}$ and $\mathcal{R} = \mathbb{Z}[\tau]$, where τ is the Frobenius endomorphism on a Koblitz curve). This is more stringent than necessary, however, it will make the proofs less elaborate.

Notation: For $\gcd(a, b) = 1$, we denote $\text{ord}_b(a)$ the multiplicative order of $a \pmod{b}$.

Lemma 1. *Let π be a prime, $p > 0$ a generator of $\pi\mathcal{R} \cap \mathbb{Z}$ and $k \geq 2$ an integer. Let $a \in \mathcal{R}$. Then $(1 + a\pi^k)^p \equiv 1 + pa\pi^k \pmod{\pi^{k+2}}$.*

Proof. Note first that p is prime in \mathbb{Z} . Using the binomial theorem, we write out the left-hand side of the congruence as $(1 + a\pi^k)^p = 1 + pa\pi^k + \sum_{i=2}^p \binom{p}{i} a^i \pi^{ki}$. If $k \geq 2$, then $2k \geq k + 2$ so that $\pi^{k+2} \mid \pi^{ki}$. □

Now the following result is proved immediately by induction.

Lemma 2. *Let π, p, k, a as in Lemma 1. If $u \geq 0$, then $(1 + a\pi^k)^{p^u} \equiv 1 + p^u a \pi^k \pmod{\pi^{k+u+1}}$.*

Lemma 3. *Let π, p be as in Lemma 1, $\alpha, \beta \in \mathcal{R}$ such that $\alpha \equiv \beta \pmod{\pi^u}$ for some $u \geq 1$. Then $\alpha^p \equiv \beta^p \pmod{\pi^{u+1}}$.*

Proof. We proceed as in the proof of Lemma 1. We write $\alpha = \beta + a\pi^u$. Then $\alpha^p = \beta^p + \sum_{i=1}^p \binom{p}{i} \beta^{p-i} a^i \pi^{iu}$. Note that $\pi^{u+1} \mid \pi^{iu}$ if $i \geq 2$. For $i = 1$ the term in the summation is $p\beta^{p-1}a\pi^u$. Since $\pi \mid p$, we are done. □

Theorem 1. *Let $\alpha \in \mathcal{R}$ and $d = \text{ord}_{\pi^2}(\alpha)$. Assume also that π is unramified over p , in other words that $\pi \nmid (p/\pi)$. Let*

$$k = \max\{u \geq 2: d = \text{ord}_{\pi^u}(\alpha)\} .$$

Then

$$\text{ord}_{\pi^u}(\alpha) = \begin{cases} d & \text{if } u \leq k , \\ dp^{u-k} & \text{if } u > k . \end{cases}$$

Proof. It is clear that $\text{ord}_{\pi^u}(\alpha) = d$ if $u \leq k$. We then prove by induction that

$$\text{ord}_{\pi^{k+u}}(\alpha) = dp^u \quad \text{if } u \geq 1 .$$

Since $\alpha^d \equiv 1 \pmod{\pi^k}$ we deduce by Lemma 3 $\alpha^{dp} \equiv 1 \pmod{\pi^{k+1}}$. Therefore $\text{ord}_{\pi^{k+1}}(\alpha) \mid dp$ but also $d \mid \text{ord}_{\pi^{k+1}}(\alpha)$ and $d \neq \text{ord}_{\pi^{k+1}}(\alpha)$ by definition of u . Hence $\text{ord}_{\pi^{k+1}}(\alpha) = dp$ and the initial step ($u = 1$) of induction is proved.

Assume therefore that $\text{ord}_{\pi^{k+u}}(\alpha) = dp^u$.

Notice also that we must then have

$$\alpha^d = 1 + a\pi^k \pmod{\pi^{k+1}} \quad \text{where } \pi \nmid a .$$

By Lemma 2, we then have

$$\alpha^{dp^u} \equiv 1 + p^u a \pi^k \equiv 1 + a(p/\pi)^u \pi^{k+u} \pmod{\pi^{k+u+1}} .$$

Since $\pi \mid p$ is unramified, we have $\alpha^{dp^u} \not\equiv 1 \pmod{\pi^{k+u+1}}$. By the induction hypothesis, $dp^u \mid \text{ord}_{\pi^{k+u+1}}(\alpha)$ and we just found that these two numbers are different. Since by Lemma 3 again $\text{ord}_{\pi^{k+u+1}}(\alpha) \mid dp^{u+1}$ and p is prime, it must be $\text{ord}_{\pi^{k+u+1}}(\alpha) = dp^{u+1}$. This completes the proof. □

We can appeal to this theorem to easily find the order of known elements to a power of a prime. We let τ be the Frobenius on a Koblitz curve as described previously, viewing it as a complex root of $X^2 + (-1)^a X + 2 = 0$. Then $\mathbb{Z}[\tau]$ is Euclidean hence a unique factorization domain. We have that τ is prime in $\mathbb{Z}[\tau]$ and likewise for $\bar{\tau} = (-1)^{a+1} - \tau$, its complex conjugate. Also, $\tau \mid 2 = \tau\bar{\tau}$ is unramified, since τ and $\bar{\tau}$ are coprime.

Corollary 1. *We have the following.*

$$\begin{aligned} \text{ord}_{3^u}(2) &= 2 \cdot 3^{u-1} & u \geq 1 , \\ \text{ord}_{2^u}(3) &= 2^{u-2} & u \geq 3 , \\ \text{ord}_{\tau^u}(3) &= 2^{u-2} & u \geq 3 , \\ \text{ord}_{\tau^u}(\bar{\tau}) &= 2^{u-2} & u \geq 3 . \end{aligned}$$

Proof. The first equality follows from the fact that $6 = \text{ord}_9(2) < \text{ord}_{27}(2)$ and an actual verification for $u = 1$.

For the second, notice that $\text{ord}_4(3) = 2 = \text{ord}_8(3) < \text{ord}_{16}(3)$.

For the third, it suffices to notice that $2^u \mid 3^i - 1$ if and only if $\tau^u \mid 3^i - 1$. The “only if part” is obvious, since $\tau \mid 2$. For the “if” part, notice that by taking conjugates we also have $\bar{\tau}^u \mid 3^i - 1$ and since τ and $\bar{\tau}$ are coprime we get $\tau^u \bar{\tau}^u \mid 3^i - 1$.

Finally, $(-1)^{a+1}\bar{\tau} = -1 - \tau^2$, hence $\bar{\tau}^2 = 1 + \bar{\tau}\tau^3 + \tau^4$. This yields immediately $2 = \text{ord}_{\tau^2}(\bar{\tau}) = \text{ord}_{\tau^3}(\bar{\tau}) < \text{ord}_{\tau^4}(\bar{\tau})$ if $a = 1$ or $1 = \text{ord}_{\tau^2}(\bar{\tau}) < 2 = \text{ord}_{\tau^3}(\bar{\tau}) < \text{ord}_{\tau^4}(\bar{\tau})$ if $a = 0$ and the last formula. \square

This leads to the main theorem of this section.

Theorem 2. *1. Every $N \in \mathbb{Z}$ with $3 \nmid a$ is congruent modulo 3^u , ($u \geq 1$), to precisely one of the numbers $2^j, 0 \leq j < 2 \cdot 3^{u-1}$.*
2. Every $N \in \mathbb{Z}[\tau]$ with $\tau \nmid N$ is congruent modulo τ^u , ($u \geq 3$), to precisely one of the numbers $(-1)^e A^j, e = 0, 1$ and $0 \leq j < 2^{u-2}$, for $A = 3$ or $\bar{\tau}$.

Proof. There are exactly $\phi(3^u) = 2 \cdot 3^{u-1}$ residue classes coprime to the modulus 3^u . Hence, the first part of the theorem follows from the first equality of Corollary 1.

For the second, begin by noting that $\#\mathbb{Z}[\tau]/\tau^u = 2^u$ (since the norm of τ^u is 2^u) and $\#(\mathbb{Z}[\tau]/\tau^u)^* = 2^{u-1}$, since elements divisible by τ are exactly the kernel of the reduction homomorphism $\mathbb{Z}[\tau]/\tau^u \rightarrow \mathbb{Z}[\tau]/\tau$. Therefore it suffices to prove that the numbers listed in the theorem are all distinct modulo τ^u . Suppose then that $(-1)^e A^j \equiv (-1)^{e'} A^{j'} \pmod{\tau^u}$. Reducing modulo τ^3 , we get that $e = e'$, since the coprime residues modulo τ^3 are $\pm 1, \pm A$. Hence $A^j \equiv A^{j'} \pmod{\tau^u}$ and by Corollary 1, we must have $j = j'$. This proves the theorem. \square

4 Algebraic Algorithms for DBNS Recoding and Scalar Multiplication

The results hitherto proved allow us to provide new double base recodings of scalars. Unlike previous algorithms [4,6,8,9] these are not greedy and proceed from right to left (i.e. from the smallest powers of the fast endomorphism to the largest).

Algorithm 1 implements a first version of a new DBNS recoding. We have given here an unsigned version, which, by a result of [4] must have at least $(1 + o(1)) \log n / \log \log n$ terms. The algorithm works by Theorem 2, which says that in Step 6 we can always find j . The termination of the algorithm is also simple here since in Step 7, N stays positive but becomes strictly smaller. A signed version, suitable for implementation on E_b , can be readily obtained and is left to the reader.

Algorithm 1. Unsigned right-to-left DBNS(2,3) recoding

Input: An integer $n > 0$ and a parameter u .

Output: Two arrays $s[], t[]$ and their common length k . The arrays are sequences of exponents in the decomposition $n = \sum_{i=0}^{k-1} 2^{s[i]} 3^{t[i]}$

```

1.   $N \leftarrow n, i \leftarrow 0, t \leftarrow 0$ 
2.   $t[] \leftarrow 0, s[] \leftarrow 0$ 
3.  while  $N \geq 4^{3^{u-1}}$  do
4.      while  $3 \mid N$  do
5.           $N \leftarrow N/3, t \leftarrow t + 1$ 
6.          Find  $0 \leq j < 3^{u-1} 2$  with  $N \equiv 2^j \pmod{3^u}$ 
7.           $N \leftarrow (N - 2^j)/3^u$ 
8.           $s[i] \leftarrow j, t[i] \leftarrow t$ 
9.           $t \leftarrow t + u, i \leftarrow i + 1$ 
10. while  $N > 0$  do
11.     while  $3 \mid N$  do
12.          $N \leftarrow N/3, t \leftarrow t + 1$ 
13.     if  $N \equiv 1 \pmod{3}$  then
14.          $N \leftarrow (N - 1)/3, s[i] \leftarrow 0$ 
15.     else
16.          $N \leftarrow (N - 2)/3, s[i] \leftarrow 1$ 
17.          $t[i] \leftarrow t, t \leftarrow t + 1, i \leftarrow i + 1$ 
18. return  $s[], t[], i$ 

```

Algorithm 2 implements a signed algorithm using a complex double base $(3, \tau)$, resp. $(\bar{\tau}, \tau)$, to be used on a Koblitz curve E_a , resp. a supersingular elliptic curve in characteristic 3.

Algorithm 2. Signed right-to-left DBNS(A, τ) recoding ($A = 3$ or $\bar{\tau}$)

Input: An integer $\zeta \in \mathbb{Z}[\tau]$ and a parameter u .

Output: Three arrays $s[], t[], e[]$ and their common length k . The arrays are sequences of exponents in the decomposition $n = \sum_{i=0}^{k-1} (-1)^{e[i]} A^{s[i]} \tau^{t[i]}$.

```

1.   $N \leftarrow \zeta, i \leftarrow 0, t \leftarrow 0$ 
2.   $t[] \leftarrow 0, s[] \leftarrow 0, e[] \leftarrow 0$ 
3.  while  $|N| \geq 2^{2^{u-1}}$  [See Remarks below]
4.      while  $\tau \mid N$  do
5.           $N \leftarrow N/\tau, t \leftarrow t + 1$ 
6.          Find  $0 \leq j < 2^{u-2}$  and  $e = 0, 1$  with  $N \equiv (-1)^e A^j \pmod{\tau^u}$ 
7.           $N \leftarrow (N - (-1)^e A^j)/\tau^u$ 
8.           $s[i] \leftarrow j, t[i] \leftarrow t, e[i] \leftarrow e$ 
9.           $t \leftarrow t + u, i \leftarrow i + 1$ 
10. while  $|N| > 0$  do
11.     while  $\tau \mid N$  do

```

```

12.            $N \leftarrow N/\tau, t \leftarrow t + 1$ 
13.   if  $N \equiv 1 \pmod{\tau^2}$  then
14.            $N \leftarrow (N - 1)/\tau^2, e[i] \leftarrow 0$ 
15.   else
16.            $N \leftarrow (N + 1)/\tau^2, e[i] \leftarrow 1$ 
17.    $t[i] \leftarrow t, t \leftarrow t + 2, i \leftarrow i + 1$ 
18.   return  $s[], t[], e[], i$ 

```

Remarks

1. In the case $A = \bar{\tau}$, we can replace the lower bound in line 3. by $2^{2^{u-3}}$.
2. To reduce the length of the expansion, it is possible to adapt u to the size of N . For instance, if $A = \bar{\tau}$, replace line 3. by
 3. **while** $|N| > 0$ **do**
and after line 5. add
 6. **while** $|N| < 2^{\frac{2^{u-2}-1}{2}}$ **do** $u \leftarrow u - 1$
Doing that, lines 10. to 17. are no longer necessary. This modification helps to save a few more additions in Algorithm 4. See Table 1.

By Theorem 2 again, the algorithm is consistent. The only point left to show is that it will terminate, namely that we have eventually $N < 2^{2^{u-1}}$, since upon entering Step 10, the algorithm computes the τ -NAF of N , hence termination is guaranteed.

Indeed notice that if $N \geq 2^{2^{u-1}}$ then

$$|(-1)^e A^j| \leq 3^j < 3^{2^{u-2}} < 4^{2^{u-2}} \leq |N| \quad (2)$$

therefore in Step 7

$$\left| \frac{N - (-1)^e A^j}{\tau^u} \right| < \frac{2|N|}{|\tau^u|} = \frac{|N|}{|\tau^{u-2}|} < |N| \quad (3)$$

since $u \geq 3$. Since $|N|^2 \in \mathbb{N}$ (it is the norm of the algebraic integer $N \in \mathbb{Z}[\tau]$), eventually $|N| < 2^{2^{u-1}}$ and the algorithm terminates.

In the case when $A = \bar{\tau}$ and the lower bound is $2^{2^{u-3}}$, we replace (2) by

$$|(-1)^e \bar{\tau}^j| \leq 2^{j/2} < 2^{2^{u-3}} \leq |N|$$

and we proceed as in (3) to show that $|N|$ diminishes. Therefore our algorithms are correct. Notice that we apply Algorithm 2 to ζ , the reduced τ -NAF of n (see Section 2.3).

After running Algorithms 1 or 2 and before Algorithm 3, that computes the scalar multiplication, we have to shuffle the indices i in the arrays $e[], s[], t[]$ so as to get $s[i+1] \geq s[i]$ for all i and $t[i+1] > t[i]$ in case $s[i+1] = s[i]$. In Algorithm 3, set $e[i] = 0$ if using an unsigned recoding.

Algorithm 3. Scalar Multiplication from a DBNS(A, B) expansion

Input: A point P on the curve E_a or E_b and the arrays $e[\cdot], s[\cdot], t[\cdot]$ of length k such that $s[i + 1] \geq s[i]$ and $t[i + 1] > t[i]$ whenever $s[i + 1] = s[i]$.

Output: The point Q on E_a or E_b such that $Q = \sum_{i=0}^{k-1} (-1)^{e[i]} A^{s[i]} B^{t[i]} P$.

1. $Q \leftarrow \mathcal{O}, i \leftarrow k - 1$
 2. $s[-1] \leftarrow 0$
 3. **while** $i \geq 0$ **do**
 4. Let $j \leq i$ be the min index with $s[j] = s[i]$
 5. $R \leftarrow (-1)^{e[i]} P$
 6. **while** $i > j$ **do**
 7. $R \leftarrow B^{t[i]-t[i-1]} R + (-1)^{e[i-1]} P$
 8. $i \leftarrow i - 1$
 9. $Q \leftarrow Q + R$
 10. $Q \leftarrow A^{s[i]-s[i-1]} Q$
 11. **return** Q
-

5 Comparison with Established Methods

We want here to give an idea of how well Algorithm 2 fares with $(\bar{\tau}, \tau)$ on Koblitz curves standardized by NIST. We compare our new multiplication algorithm with the τ -and-add using a τ -NAF expansion [24] and the width- w τ -NAF expansion [25].

For a given value of u , by (3), the number of iterations in the main loop (Steps 3 to 9) is bounded by the quantity c such that $|\zeta| = |\tau^{u-2}|^c = 2^{\frac{c}{2}(u-2)}$. This gives

$$c = \frac{2 \log_2 |\zeta|}{u - 2} = \frac{\mathbf{p}}{u - 2}$$

for a generic scalar, by the way ζ is constructed. Also, since the “tail” (i.e. the quantity processed in Steps 11 to 17) is a generic integer of $\mathbb{Z}[\tau]$ of norm less than $2^{2^{u-2}}$, its expected Hamming weight is bounded by $2^{u-2}/3$. Thus, the average Hamming weight of the new expansion is bounded by

$$\frac{\mathbf{p}}{u - 2} + \frac{2^{u-2}}{3} ,$$

and its worst case by

$$\frac{\mathbf{p}}{u - 2} + 2^{u-3} + 1 . \tag{4}$$

In practice, when N is large in (3), the new value of N has absolute value much closer to $|N|/|\tau^u|$, therefore we should expect a Hamming weight closer to the value

$$\frac{\mathbf{p}}{u} + \frac{2^{u-2}}{3} . \tag{5}$$

Algorithm 3 then implies that the total cost of a scalar multiplication equals at most $\mathbf{p}/u + 2^{u-2}/3$ additions plus 2^{u-2} applications of $\bar{\tau}$. Since an application

of $\bar{\tau} = (-1)^{1-a} - \tau$ corresponds to a curve addition, the total cost (in curve additions) is bounded from above by

$$f(u) = \frac{\mathbf{p}}{u} + \frac{2^u}{3} .$$

In the previous argument, following [4, Section 4], we neglected the cost of applying τ , as we will in the following comparisons. See also Section 7 for a concrete approach to reducing the impact of the Frobenius to a non-dominant term.

We can modify Algorithm 3 to make use of the advantage of halvings over multiplications by $A = \bar{\tau}$ (at least a 50% saving in performance). Indeed, let $\zeta' = 2^{2^{u-2}} \zeta \pmod{\frac{\tau^{\mathbf{p}-1}}{\tau-1}}$ with minimal norm. From a DBNS($\bar{\tau}, \tau$) expansion

$$\zeta' = \sum_{i=0}^{k-1} (-1)^{e'_i} \bar{\tau}^{s'_i} \tau^{t'_i}$$

get that

$$\begin{aligned} nP = \zeta P &= \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\bar{\tau}^{s'_i}}{2^{2^u}} \tau^{t'_i} P = \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\tau^{t'_i - s'_i}}{2^{2^u - s'_i}} P \\ &= \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\tau^{\epsilon_i \mathbf{p} + t'_i - s'_i}}{2^{2^u - s'_i}} P \end{aligned}$$

where $\epsilon_i = 1$ if $t'_i < s'_i$ and 0 else. Note that this is a valid DBNS($1/2, \tau$) expansion, because for different values of i, j , the same powers of $1/2$ and τ occur only if $s'_i = s'_j$ and either $t'_i - s'_i = t'_j - s'_j$ or $t'_i - s'_i = \mathbf{p} + t'_j - s'_j$. Since the pairs (s'_i, t'_i) arise from a DBNS expansion and $t'_i < \mathbf{p}$, either case is impossible.

In this case, from (5) and the subsequent analysis, we can conclude that the cost of one scalar multiplication using a DBNS($1/2, \tau$) expansion is upper bounded on average by $g(u)$ curve additions, where

$$g(u) = \frac{\mathbf{p}}{u} + \frac{5}{24} 2^u .$$

For various parameters of \mathbf{p} corresponding to the NIST curves K-163 ($a = 1$), K-233 ($a = 0$), K-283 ($a = 0$), K-409 ($a = 0$), K-571 ($a = 0$), Table 1 gives the scalar multiplication costs in elliptic curve additions (with the assumption that two halvings are equivalent to one addition) using the τ -NAF, width- w τ -NAF (w - τ -NAF) and our new recodings, on average, as well as the percentage improvement over those methods and the value of u used in minimizing the functions $f(u)$ and $g(u)$. In each case, the average is computed over 25,000 values.

6 Asymptotic Improvements

We now establish the asymptotic behavior of our new scalar multiplication algorithm. Its sublinear nature will be thus revealed. We have the following.

Table 1. Comparison of scalar multiplication algorithms on Koblitz curves

Field size \mathbf{p}	τ -NAF	w - τ -NAF	w	DBNS($\bar{\tau}, \tau$)	u	DBNS($\frac{1}{2}, \tau$)	u	%/ τ -NAF	%/ w - τ -NAF
163	54.33	34.16	5	34.60	5	31.09	5	42.78%	8.99%
233	77.66	45.83	5	46.60	5	41.38	6	46.72%	9.71%
283	94.33	54.16	5	54.38	5	48.80	6	48.27%	9.90%
409	136.33	73.42	6	74.40	6	66.89	6	50.94%	8.90%
571	190.33	102.37	6	97.18	6	88.04	7	53.74%	14.00%

Theorem 3. Algorithms 1 and 2 allow to express nP , where $P \in E_b$ or $P \in E_a$, as

$$nP = \left(\sum_{i=0}^{k-1} (-1)^{e_i} A^{s_i} B^{t_i} \right) P \quad \text{with } (s_i, t_i) \neq (s_j, t_j) \text{ for } i \neq j ,$$

where $(A, B) = (2, 3)$ in the case of E_b and $(A, B) = (3, \tau)$ or $(\bar{\tau}, \tau)$ in the case of E_a . The length k satisfies on average (the worst case being twice as large only in the case of E_a)

$$k \leq (1 + o(1)) \frac{\log n}{\log \log n} \quad \text{as } n \rightarrow \infty ,$$

and $\max s_i \leq \log n / (\log \log n)^2$.

Therefore scalar multiplication nP can be performed via Algorithm 3 on these curves with an average cost of less than $(1 + o(1)) \log n / \log \log n$ curve additions.

Proof. We detail the proof in the case of Koblitz curves. In the DBNS(2, 3) case, simple modifications lead to the analogous result. We start with (4), letting $u = \lfloor 2 + \log_2 \mathbf{p} - 2 \log_2 \log \mathbf{p} \rfloor$. We then find that $k \leq \frac{\mathbf{p}}{\log_2 \mathbf{p}} + o\left(\frac{\mathbf{p}}{\log \mathbf{p}}\right)$. Since on average $\mathbf{p} = \log_2 n$ we are done in the average case. In the worst case \mathbf{p} has to be replaced by $2 \log_2 |\zeta|$, where $\zeta = n$ if n is too small. The (average) bound on the s_i is immediate from Step 6 in Algorithm 2.

Since the total cost of Algorithm 3 differs from the Hamming weight k by a multiple of $2^{u-2} = o(\mathbf{p} / \log \mathbf{p})$ we are done. \square

7 On the Use of Normal vs. Polynomial Bases

Neglecting the cost of τ is fine if normal bases are used, but when polynomial bases are used Frobenius operations can become expensive as u increases. One solution is provided, as already mentioned, by a technique introduced by Park et al. in [20] and used by Okeya et al. in [19]. Instead of applying a variable power of the Frobenius to a changing point as done in Steps 5 to 9 if Algorithm 3, we apply the Frobenius to the point P and accumulate directly. Only, the Frobenius is performed on a copy of P that has been converted to normal basis representation (hence, all powers of the Frobenius have essentially the same cost), and then the result is converted back to polynomial basis representation before adding it to the accumulator variable that will contain the final result at the end of the algorithm.

Algorithm 4. $(\bar{\tau}, \tau)$ -Double Bases Scalar Multiplication on Koblitz Curves

Input: A point P on E_a , a scalar z and arrays $e[\cdot], s[\cdot], t[\cdot]$ of length k with $s[i+1] \geq s[i]$ such that $z = \sum_{i=0}^{k-1} (-1)^{e[i]} \bar{\tau}^{s[i]} \tau^{t[i]}$.

Output: The point Q on E_a such that $Q = zP = \sum_{i=0}^{k-1} (-1)^{e[i]} \bar{\tau}^{s[i]} \tau^{t[i]} P$.

1. $R \leftarrow \text{normal_basis}(P)$ [Keep in affine coordinates]
 2. $Q \leftarrow 0$ [Use López-Dahab coordinates]
 3. **for** $i = k - 1$ **to** 0 **do**
 4. **if** $i \neq k - 1$ **and** $s[i] \neq s[i + 1]$ **then**
 5. **for** $j = 1$ **to** $s[i + 1] - s[i]$ **do**
 6. $Q \leftarrow \tau^{-1}Q, Q \leftarrow 2Q$
 7. $Q \leftarrow Q + e[i] \cdot \text{polynomial_basis}(\tau^{t[i]}R)$ [Mixed coordinates]
 8. **return** Q
-

With our notation the resulting method is presented as Algorithm 4, in a version that uses mixed coordinate arithmetic and projective (\mathcal{P}) or López-Dahab (\mathcal{LD}) coordinates [12, § 13.3] while keeping the points P and R in affine (\mathcal{A}) coordinates.

There we use the fact that $2 = \tau\bar{\tau}$ to implement $\bar{\tau}$ as a doubling with an inverse of a Frobenius, an operation that requires three square root extractions in \mathcal{P} or \mathcal{LD} . A square root extraction costs between 1/8 and 1/2 of a multiplication depending on the field [14]. A doubling in \mathcal{LD} costs 4 multiplications and 4 squarings, whereas a mixed coordinate addition (i.e. adding a point in \mathcal{A} to a point in \mathcal{LD} with a result in \mathcal{LD}) costs 9 multiplications and 5 squarings. The time required by a basis conversion (routines `normal_basis` and `polynomial_basis`) is roughly the same as one polynomial basis multiplication, and the conversion routines require each a matrix that occupies $O(\mathbf{p}^2)$ bits of storage [7]. Hence Steps 1 and 6 cost each about two field multiplications. The time for an evaluation of $\bar{\tau}$ is then roughly a half of the time for an evaluation of the addition (including the basis conversion).

8 Conclusion

This work shows that using double bases in scalar multiplication improves performance significantly, even for the smallest cryptographic parameters, at almost no additional memory cost. This method however is only effective if multiplication by one of the bases can be neglected, as was shown in [4]. The resulting new scalar multiplication algorithms are especially fast on Koblitz curves and supersingular curves of characteristic three used in pairing-based cryptosystems.

As this work is being written, other articles on the same subject are about to be published. In [10], accepted at CHES 2006, the authors present practical measurements on FPGA and show that indeed one achieves a 50% speedup already on the smallest Koblitz curve K-163 by using short decompositions found by a clever extensive search. The paper [2], to appear in the proceedings of SAC 2006,

among other things contains results similar to ours, but expressed in the language of expansions with respect to a single base using suitably defined digit sets.

References

1. R. Avanzi, M. Ciet, and F. Sica. *Faster Scalar Multiplication on Koblitz Curves combining Point Halving with the Frobenius Endomorphism*. In *Proceedings of PKC 2004*, Lecture Notes in Computer Science 2947, pp. 28–40. Springer, 2004.
2. R. Avanzi, C. Heuberger, and H. Prodinger. *On Redundant τ -adic Expansions and Non-Adjacent Digit Sets*. To appear in *Proceedings of SAC 2006 (Workshop on Selected Areas in Cryptography)*, Lecture Notes in Computer Science. Springer.
3. R. Avanzi, C. Heuberger, and H. Prodinger. *Minimality of the Hamming Weight of the τ -NAF for Koblitz Curves and Improved Combination with Point Halving*. In *Proceedings of SAC 2005*, Lecture Notes in Computer Science 3897, pp. 332–344. Springer, 2006.
4. R. Avanzi and F. Sica. *Scalar Multiplication on Koblitz Curves using Double Bases*. To appear in *Proceedings of Vietcrypt 2006*, Lecture Notes in Computer Science. Springer, 2006.
5. P. Barreto, H. Y. Kim, B. Lynn, and M. Scott. *Efficient algorithms for pairing-based cryptosystems*. In *Advances in Cryptology - CRYPTO 2002*, Lecture Notes in Computer Science 2442, pp. 354–369. Springer, 2002.
6. M. Ciet and F. Sica. *An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm*. In *Progress in Cryptology - Proceedings of Mycrypt 2005*, Lecture Notes in Computer Science 3715, pp. 171–182. Springer, 2005.
7. J.-S. Coron, D. M'Raihi, and C. Tymen. *Fast generation of pairs $(k, [k]P)$ for Koblitz elliptic curves*. In *Proceedings of SAC 2001*, Lecture Notes in Computer Science 2259, pp. 151–164. Springer, 2001.
8. V. S. Dimitrov, L. Imbert, and P. K. Mishra. *Efficient and secure elliptic curve point multiplication using double-base chains*. In *Advances in Cryptology - ASIACRYPT 2005*, Lecture Notes in Computer Science 3788, pp. 59–78. Springer, 2005.
9. V. S. Dimitrov, L. Imbert, and P. K. Mishra. *Fast elliptic curve point multiplication using double-base chains*. Cryptology ePrint Archive, Report 2005/069, 2005. Available from <http://eprint.iacr.org/>.
10. V. S. Dimitrov, K. Jarvinen, M. J. Jacobson Jr, W. F. Chan, and Z. Huang. *FPGA Implementation of Point Multiplication on Koblitz Curves Using Kleinian Integers*. In *Proceedings of CHES 2006*, Lecture Notes in Computer Science. Springer, 2006.
11. V. S. Dimitrov, G. A. Jullien, and W. C. Miller. *An algorithm for modular exponentiation*. *Information Processing Letters*, 66(3):155–159, 1998.
12. C. Doche and T. Lange. *Arithmetic of Elliptic Curves, in Handbook of Elliptic and Hyperelliptic Curve Cryptography, H. Cohen and G. Frey Eds*. CRC Press, Inc., 2005.
13. C. Doche and T. Lange. *Arithmetic of Special Curves, in Handbook of Elliptic and Hyperelliptic Curve Cryptography, H. Cohen and G. Frey Eds*. CRC Press, Inc., 2005.
14. K. Fong, D. Hankerson, J. López, and A. J. Menezes. *Field Inversion and Point Halving Revisited*. *IEEE Trans. Comp.*, 53(8), pp. 1047–1059, August 2004.
15. R. P. Gallant, J. L. Lambert, and S. A. Vanstone. *Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms*. In *Advances in Cryptology - CRYPTO 2001*, Lecture Notes in Computer Science 2139, pp. 190–200. Springer, 2001.

16. E. W. Knudsen. *Elliptic Scalar Multiplication Using Point Halving*. In *Advances in Cryptography - ASIACRYPT 1999*, Lecture Notes in Computer Science 1716, pp. 135–149. Springer, 1999.
17. N. Koblitz. *CM-curves with good cryptographic properties*. In *Advances in Cryptology - CRYPTO 1991*, Lecture Notes in Computer Science 576, pp. 279–287, Berlin, 1991. Springer.
18. A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
19. K. Okeya, T. Takagi, and C. Vuillaume. *Short Memory Scalar Multiplication on Koblitz Curves*. In *Proceedings of CHES 2005*, Lecture Notes in Computer Science 3659, pp. 91–105. Springer, 2005.
20. D. J. Park, S. G. Sim, and P. J. Lee. *Fast scalar multiplication method using change-of-basis matrix to prevent power analysis attacks on Koblitz curves*. In *Proceedings of WISA 2003*, Lecture Notes in Computer Science 2908, pp. 474–488. Springer, 2003.
21. G.W. Reitwiesner. *Binary arithmetic*. *Advances in Computers*, 1, pp. 231–308, 1960.
22. R. Schroepfel. *Elliptic curve point ambiguity resolution apparatus and method*. International Application Number PCT/US00/31014, filed 9 November 2000.
23. R. Schroepfel. *Elliptic curves: Twice as fast!*, 2000. Presentation at the Crypto 2000 Rump Session.
24. J. A. Solinas. *An Improved Algorithm for Arithmetic on a Family of Elliptic Curves*. In *Advances in Cryptology - CRYPTO 1997*, Lecture Notes in Computer Science 1294, pp. 357–371. Springer, 1997.
25. J. A. Solinas. *Efficient arithmetic on Koblitz curves*. *Designs, Codes and Cryptography*, 19, pp. 195–249, 2000.

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

HIBE With Short Public Parameters Without Random Oracle

Sanjit Chatterjee and Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108
{sanjit_t, palash}@isical.ac.in

Abstract. At Eurocrypt 2005, Waters presented an identity based encryption (IBE) protocol which is secure in the full model without random oracle. In this paper, we extend Waters' IBE protocol to a hierarchical IBE (HIBE) protocol which is secure in the full model without random oracle. The only previous construction in the same setting is due to Waters. Our construction improves upon Waters' HIBE by significantly reducing the number of public parameters.

1 Introduction

The concept of identity based encryption (IBE) was introduced by Shamir in 1984 [17]. An IBE is a type of public key encryption where the public key can be any binary string. The corresponding secret key is generated by a private key generator (PKG) and provided to the legitimate user. The notion of IBE simplifies several applications of public key cryptography. The first efficient implementation and an appropriate security model for IBE was provided by Boneh and Franklin [5].

The PKG issues a private key associated with an identity. The notion of hierarchical identity based encryption (HIBE) was introduced in [14,13] to reduce the workload of the PKG. An entity in a HIBE structure has an identity which is a tuple (v_1, \dots, v_j) . The private key corresponding to such an identity can be generated by the entity whose identity is (v_1, \dots, v_{j-1}) and which possesses the private key corresponding to his identity. The security model for IBE was extended to that of HIBE in [14,13].

The construction of IBE in [5] and of HIBE in [13], was proved to be secure in appropriate models using the *random oracle* heuristic, i.e., the protocols make use of cryptographic hash functions that are modeled as random oracle in the security proof. The first construction of an IBE which can be proved to be secure in the full model without the random oracle heuristic was given by Boneh and Boyen in [3]. Later, Waters [19] presented an efficient construction of an IBE which is secure in the same setting.

An important construction of a HIBE is given by Boneh-Boyen [2]. This paper describes a general framework for constructing a HIBE. For an h -level HIBE,

the idea in [2] is to use h functions ψ_1, \dots, ψ_h , where ψ_i is viewed as a hash function which maps the i th component of the identity tuple to an appropriate group element. This framework is instantiated in [2] to obtain a HIBE protocol which can be proved secure in weaker model called the selective-ID (sID) model.

The construction by Waters in [19] can be viewed as another instantiation of a 1-level BB-framework [2]. Identities are considered to be n -bit strings. The construction uses group elements U', U_1, \dots, U_n (and P, P_1, P_2) as public parameters. A natural extension of this construction to an h -level HIBE is given in [19]. In this extension, for an h -level HIBE, the public parameters will be of the form $U'_1, U_{1,1}, \dots, U_{1,n}, U'_2, U_{2,1}, \dots, U_{2,n}, \dots, U'_h, U_{h,1}, \dots, U_{h,n}$. One still requires the parameters P, P_1, P_2 , giving rise to $3 + (n + 1)h$ many parameters.

OUR CONTRIBUTIONS: We present a HIBE which can be proved to be secure in the full model assuming the decisional bilinear Diffie-Hellman problem to be hard without using the random oracle heuristic. Our construction can also be viewed as another instantiation of the BB-framework [2]. The public parameters for an h -level HIBE are of the form $U'_1, \dots, U'_h, U_1, \dots, U_n$. In other words, the parameters U'_1, \dots, U'_h correspond to the different levels of the HIBE, whereas the parameters U_1, \dots, U_n are the same for all the levels. These parameters U_1, \dots, U_n are reused in the key generation procedure. We require $3 + n + h$ parameters compared to $3 + (n + 1)h$ parameters in Waters' HIBE.

The reuse of public parameters over the different levels of the HIBE complicates the security proof. A straightforward extension of the independence results and lower bound proofs from [19] is not possible. We provide complete proofs of the required results. The constructed HIBE is proved to be secure under chosen plaintext attack (called CPA-secure). Standard techniques [8,6] can convert such a HIBE into one which is secure against chosen ciphertext attack (CCA-secure).

RELATED WORK: The first construction of HIBE which is secure in the full model is due to Gentry and Silverberg [13]. The security proof depends on the random oracle heuristic. HIBE constructions which can be proved secure without random oracle are known [2,4]. However, these are secure in the weaker selective-ID model. A generic transformation converts a selective-ID secure HIBE to a HIBE secure in the full model. Unfortunately, this results in an unacceptable degradation in the security bound. It is also possible to convert it into a HIBE secure in the full model *under* the random oracle hypothesis. As mentioned earlier, Waters [19] HIBE is the only previous indication of directly obtaining a HIBE which is secure in the full model without random oracle. In Table 1 of Section 4, we provide a comparison of our construction with the previous constructions.

An extension of Waters' IBE was independently done by Chatterjee-Sarkar [9] and Naccache [16]. In this extension, the n -bit identities of Waters' IBE are replaced by l strings of length n/l bits each. This reduces the number of public parameters from $3 + n$ in Waters' IBE to $3 + l$. The trade-off is a further security degradation by a factor of approximately $2^{n/l}$. This can be translated into a trade-off between the size of the public parameters and the efficiency of the protocol (see [9]). The CSN idea of extending Waters' IBE can also be applied to the HIBE we describe.

2 Definitions

In this section, we describe HIBE, security model for HIBE, cryptographic bilinear map and the hardness assumption that will be required in the proof.

2.1 HIBE Protocol

Following [14,13] a HIBE scheme is specified by four probabilistic algorithms: Setup, Key Generation, Encryption and Decryption. Note that, for a HIBE of height h (henceforth denoted as h -HIBE) any identity \mathbf{v} is a tuple (v_1, \dots, v_j) where $1 \leq j \leq h$.

Setup: It takes as input a security parameter and returns the system parameters together with the master key. The system parameters include the public parameters of the PKG, a description of the message space, the ciphertext space and the identity space. These are publicly known while the master key is known only to the PKG.

Each of the algorithms below (Key Generation, Encryption and Decryption) have the system public parameters as an input. We do not mention this explicitly.

Key Generation: It takes as input an identity $\mathbf{v} = (v_1, \dots, v_j)$, the public parameters of the PKG and the private key $d_{\mathbf{v}_{|(j-1)}}$ corresponding to the identity (v_1, \dots, v_{j-1}) and returns a private key $d_{\mathbf{v}}$ for \mathbf{v} . The identity \mathbf{v} is used as the public key while $d_{\mathbf{v}}$ is the corresponding private key. If $j = 1$, then the private key is generated by the PKG. It is not difficult to see that any entity which possesses a private key for a prefix of \mathbf{v} can generate a private key for \mathbf{v} .

Encryption: It takes as input the identity \mathbf{v} , the public parameters of the PKG and a message from the message space and produces a ciphertext in the ciphertext space.

Decryption: It takes as input the ciphertext and the private key of the corresponding identity \mathbf{v} and returns the message or **bad** if the ciphertext is not valid.

2.2 Security Model for HIBE

Security is defined using an adversarial game. An adversary \mathcal{A} is allowed to query two oracles – a decryption oracle and a key-extraction oracle. At the initiation, it is provided with the public parameters of the PKG. The game has two query phases with a challenge phase in between.

Query Phase 1: Adversary \mathcal{A} makes a finite number of queries where each query is addressed either to the decryption oracle or to the key-extraction oracle. In a query to the decryption oracle it provides a ciphertext as well as the identity under which it wants the decryption. It gets back the corresponding message or **bad** if the ciphertext is invalid. Similarly, in a query to the key-extraction oracle, it asks for the private key of the identity it provides and gets back this private key. Further, \mathcal{A} is allowed to make these queries adaptively, i.e., any query may

depend on the previous queries as well as their answers. The adversary is not allowed to make any useless queries, i.e., queries for which it can compute the answer itself. For example, the adversary is not allowed to ask for the decryption of a message under an identity if it has already obtained a private key corresponding to the identity.

Challenge: At this stage, \mathcal{A} outputs an identity $\mathbf{v}^* = (v_1^*, \dots, v_j^*)$ for $1 \leq j \leq h$, and a pair of messages M_0 and M_1 . There is the natural restriction on the adversary, that it cannot query the key extraction oracle on \mathbf{v}^* or any of its proper prefixes in either of the phases 1 or 2. A random bit b is chosen and the adversary is provided with C^* which is an encryption of M_b under \mathbf{v}^* .

Query Phase 2: \mathcal{A} now issues additional queries just like Phase 1, with the (obvious) restrictions that it cannot ask the decryption oracle for the decryption of C^* under \mathbf{v}^* , nor the key-extraction oracle for the private key \mathbf{v}^* or any of its prefix.

Guess: \mathcal{A} outputs a guess b' of b .

The advantage of the adversary \mathcal{A} is defined as:

$$\text{Adv}_{\mathcal{A}}^{\text{HIBE}} = |\Pr[(b = b')] - 1/2|.$$

The quantity $\text{Adv}_{\mathcal{A}}^{\text{HIBE}}(t, q_{\text{D}}, q_{\text{C}})$ denotes the maximum of $\text{Adv}_{\mathcal{A}}^{\text{HIBE}}$ where the maximum is taken over all adversaries running in time at most t and making at most q_{C} queries to the decryption oracle and at most q_{D} queries to the key-extraction oracle. A HIBE protocol is said to be $(\epsilon, t, q_{\text{D}}, q_{\text{C}})$ -CCA secure if $\text{Adv}_{\mathcal{A}}^{\text{HIBE}}(t, q_{\text{D}}, q_{\text{C}}) \leq \epsilon$.

In the above game, we can restrict the adversary \mathcal{A} from querying the decryption oracle. $\text{Adv}_{\mathcal{A}}^{\text{HIBE}}(t, q)$ in this context denotes the maximum advantage where the maximum is taken over all adversaries running in time at most t and making at most q queries to the key-extraction oracle. A HIBE protocol is said to be (t, q, ϵ) -CPA secure if $\text{Adv}_{\mathcal{A}}^{\text{HIBE}}(t, q) \leq \epsilon$.

As mentioned earlier there are generic techniques [8,6] for converting a CPA-secure HIBE into a CCA-secure HIBE. In view of these techniques, we will concentrate only on CPA-secure HIBE.

2.3 Cryptographic Bilinear Map

Let G_1 and G_2 be cyclic groups having the same prime order p and $G_1 = \langle P \rangle$, where we write G_1 additively and G_2 multiplicatively. A mapping $e : G_1 \times G_1 \rightarrow G_2$ is called a cryptographic bilinear map if it satisfies the following properties.

- Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in \mathbb{Z}_p$.
- Non-degeneracy: If $G_1 = \langle P \rangle$, then $G_2 = \langle e(P, P) \rangle$.
- Computability: There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Since $e(aP, bP) = e(P, P)^{ab} = e(bP, aP)$, $e(\cdot)$ also satisfies the symmetry property. The modified Weil pairing [5] and the modified Tate pairing [1,11] are examples of cryptographic bilinear maps.

Note: Known examples of $e()$ have G_1 to be a group of Elliptic Curve (EC) points and G_2 to be a subgroup of a multiplicative group of a finite field. Hence, in papers on pairing implementations [1,11], it is customary to write G_1 additively and G_2 multiplicatively. On the other hand, some “pure” protocol papers [2,3,19] write both G_1 and G_2 multiplicatively though this is not true for the initial protocol papers [15,5]. Here we follow the first convention as it is closer to the known examples of cryptographic bilinear map.

The decisional bilinear Diffie-Hellman (DBDH) problem in $\langle G_1, G_2, e \rangle$ [5] is as follows: Given a tuple $\langle P, aP, bP, cP, Z \rangle$, where $Z \in G_2$, decide whether $Z = e(P, P)^{abc}$ (which we denote as Z is real) or Z is random.

The advantage of a probabilistic algorithm \mathcal{B} , which takes as input a tuple $\langle P, aP, bP, cP, Z \rangle$ and outputs a bit, in solving the DBDH problem is defined as

$$\text{Adv}_{\mathcal{B}}^{\text{DBDH}} = |\Pr[\mathcal{B}(P, aP, bP, cP, Z) = 1 | Z \text{ is real}] - \Pr[\mathcal{B}(P, aP, bP, cP, Z) = 1 | Z \text{ is random}]|$$

where the probability is calculated over the random choices of $a, b, c \in \mathbb{Z}_p$ as well as the random bits used by \mathcal{B} . The quantity $\text{Adv}_{\mathcal{B}}^{\text{DBDH}}(t)$ denotes the maximum of $\text{Adv}_{\mathcal{B}}^{\text{DBDH}}$ where the maximum is taken over all adversaries \mathcal{B} running in time at most t . By the (ϵ, t) -DBDH assumption we mean $\text{Adv}^{\text{DBDH}}(t) \leq \epsilon$.

3 HIBE Construction

The IBE scheme proposed in [19] has some similarities with the 1-level (H)IBE scheme of Boneh-Boyen [2]. Waters in his paper [19], utilized this similarity to build a HIBE in an obvious manner, i.e., for each level we have to generate new parameters. This makes the public parameters quite large – for a HIBE of height h with n -bit identities, the number of public parameters becomes $n \times h$.

Here we present an alternative construction where the public parameters can be significantly reduced. We show that for an h -HIBE it suffices to store $(n + h)$ elements in the public parameter.

The identities are of the type $(\mathbf{v}_1, \dots, \mathbf{v}_j)$, for $j \in \{1, \dots, h\}$ where each $\mathbf{v}_k = (\mathbf{v}_{k,1}, \dots, \mathbf{v}_{k,n})$, $\mathbf{v}_{k,j} \in \{0, 1\}$ for $1 \leq j \leq n$.

Let G_1 and G_2 be cyclic groups having the same prime order p . We use a cryptographic bilinear map $e : G_1 \times G_1 \rightarrow G_2$ the definition of which is given in Section 2.3. The message space is G_2 .

Set-Up: The protocol is built from groups G_1, G_2 and a bilinear map e as mentioned above. The public parameters are the following elements: $P, P_1 = \alpha P, P_2, U'_1, \dots, U'_h, U_1, \dots, U_n$, where $G_1 = \langle P \rangle$, α is chosen randomly from \mathbb{Z}_p and the other quantities are chosen randomly from G_1 . The master secret is αP_2 . (The quantities P_1 and P_2 are not directly required; instead $e(P_1, P_2)$ is required. Hence one may store $e(P_1, P_2)$ as part of the public parameters instead of P_1 and P_2 .)

Note that for the j th level of the HIBE, we add a single element, i.e., U'_j in the public parameter while the elements U_1, \dots, U_n are re-used for each level. This way we are able to shorten the public parameter size.

A shorthand: Let $v = (v_1, \dots, v_n)$, where each v_i is a bit. For $1 \leq k \leq h$ we define,

$$V_k(v) = U'_k + \sum_{i=1}^n v_i U_i. \tag{1}$$

When v is clear from the context we will write V_k instead of $V_k(v)$. The modularity introduced by this notation allows an easier understanding of the protocol.

Key Generation: Let $\mathbf{v} = (v_1, \dots, v_j)$, $j \leq h$, be the identity for which the private key is required. The private key $d_{\mathbf{v}}$ for \mathbf{v} is defined to be a tuple $d_{\mathbf{v}} = (d_0, d_1, \dots, d_j)$ where

$$d_0 = \alpha P_2 + \sum_{k=1}^j r_k V_k(\mathbf{v}_k); \text{ and } d_k = r_k P \text{ for } 1 \leq k \leq j.$$

Here r_1, \dots, r_j are random elements from \mathbb{Z}_p .

Such a key can be generated by an entity which possesses a private key for the tuple (v_1, \dots, v_{j-1}) in the manner shown in [2]. Suppose $(d'_0, d'_1, \dots, d'_{j-1})$ is a private key for the identity (v_1, \dots, v_{j-1}) . To generate a private key for \mathbf{v} , first choose a random $r_j \in \mathbb{Z}_p$ and compute $d_{\mathbf{v}} = (d_0, d_1, \dots, d_j)$ as follows.

$$d_0 = d'_0 + r_j V_j(\mathbf{v}_j); \ d_i = d'_i \text{ for } 1 \leq i \leq j - 1; \text{ and } d_j = r_j P.$$

In fact, any prefix of \mathbf{v} as well as the PKG can generate a private key $d_{\mathbf{v}}$ for \mathbf{v} .

Encryption: Let $\mathbf{v} = (v_1, \dots, v_j)$ be the identity under which a message $M \in G_2$ is to be encrypted. Choose t to be a random element of \mathbb{Z}_p . The ciphertext is

$$(C_0 = M \times e(P_1, P_2)^t, C_1 = tP, B_1 = tV_1(\mathbf{v}_1), \dots, B_j = tV_j(\mathbf{v}_j)).$$

Decryption: Let $C = (C_0, C_1, B_1, \dots, B_j)$ be a ciphertext and the corresponding identity $\mathbf{v} = (v_1, \dots, v_j)$. Let (d_0, d_1, \dots, d_j) be the decryption key corresponding to the identity \mathbf{v} . The decryption steps are as follows.

Verify whether C_0 is in G_2 , C_1 and the B_i 's are in G_1 . If any of these verifications fail, then return **bad**, else proceed with further decryption as follows. Compute $V_1(\mathbf{v}_1), \dots, V_j(\mathbf{v}_j)$. Return

$$C_0 \times \frac{\prod_{k=1}^j e(B_k, d_k)}{e(d_0, C_1)}.$$

It is standard to verify the consistency of decryption.

Chatterjee-Sarkar-Naccache Extension: Following [9,16], let l be a size parameter which divides n . An identity is a tuple (v_1, \dots, v_j) , $j \leq h$, where each \mathbf{v}_k , $1 \leq k \leq j$ is represented as $\mathbf{v}_k = (v_{k,1}, \dots, v_{k,l})$ where $v_{k,i}$ is an (n/l) -bit string considered to be an element of $\mathbb{Z}_{2^{n/l}}$.

The public parameters are $P, P_1, P_2, U_1, \dots, U_l$ and U'_1, \dots, U'_h . In this case, we change the definition of $V_k()$ to the following: $V_k(v) = U'_k + \sum_{i=1}^l v_i U_i$ where each v_i is a bit string of length n/l . Using this modified definition of $V_k()$ for $1 \leq k \leq h$, the key generation, encryption and decryption algorithms of the HIBE described above can be extended to the Chatterjee-Sarkar-Naccache settings.

4 Security

In this section, we state the result on security and discuss its implications. The proof is given in Section 5.

Theorem 1. *The HIBE protocol described in Section 3 is (ϵ_{hibe}, t, q) -CPA secure assuming that the (t', ϵ_{dbdh}) -DBDH assumption holds in $\langle G_1, G_2, e \rangle$, where $\epsilon_{hibe} \leq 2\epsilon_{dbdh}/\lambda$; $t' = t + \chi(\epsilon_{hibe})$ and*

$$\begin{aligned} \chi(\epsilon) &= O(\tau q + O(\epsilon^{-2} \ln(\epsilon^{-1}) \lambda^{-1} \ln(\lambda^{-1}))); \\ \tau &\text{ is the time required for one scalar multiplication in } G_1; \\ \lambda &= 1/(2(4q(n+1))^h). \end{aligned}$$

We further assume $4q(n+1) < p$.

The last assumption is practical and similar assumptions are also made in [19,9,16], though not quite so explicitly. Before proceeding to the proof, we discuss the above result. The main point of the theorem is the bound on ϵ_{hibe} . This is given in terms of λ and in turn in terms of q, n and h .

The reduction is not tight; security degrades by a factor of $4(4q(n+1))^h$. The actual value of degradation depends on the value of q , the number of key extraction queries made by the adversary. A value of q used in earlier analysis is $q = 2^{30}$ [12].

$h = 1$: This implies that the HIBE is actually an IBE. This is the situation originally considered by Waters [19] and $\epsilon_{hibe} \leq 16q(n+1)\epsilon_{dbdh} \leq 32nq\epsilon_{dbdh}$.

$h > 1$: This corresponds to a proper HIBE and we obtain $\epsilon_{hibe} \leq 4(4q(n+1))^h \epsilon_{dbdh} \leq 4(8nq)^h \epsilon_{dbdh}$. For $n = 160$ (and $q = 2^{30}$), this amounts to $\epsilon_{hibe} \leq 4(10 \times 2^{37})^h \epsilon_{dbdh}$.

In Table 1, we compare the known HIBE protocols which are secure in the full model. We note that HIBE protocols which are secure in the selective-ID model are also secure in the full model with a security degradation of $\approx 2^{nh}$, where h is the number of levels in the HIBE and n is number of bits in the identity. This degradation is far worse than the protocols in Table 1.

The BB-HIBE in Table 1 is obtained through a generic transformation (as mentioned in [2]) of the selective-ID secure BB-HIBE to a HIBE secure in the full model using random oracle. For the GS-HIBE [13] and BB-HIBE, the parameter q_H stands for the total number of random oracle queries and in general $q_H \approx 2^{60} \gg q$ [12]. The parameter j in the private key size, ciphertext size and the encryption and decryption columns of Table 1 represents the number of levels

Table 1. Comparison of HIBE Protocols

Protocol	Hardness Assump.	Rnd. Ora.	Sec. Deg.	Pub. Para. sz (elts. of G_1)	Pvt. Key sz (elts. of G_1)	Cprtxt sz (elts. of G_1)	Pairing	
							Enc.	Dec.
GS [13]	BDH	Yes	$q_H q^h$	2	j	j	1	j
BB [2]	DBDH	Yes	q_H^h	$h + 3$	$j + 1$	$j + 1$	None	$j + 1$
Waters [19]	DBDH	No	$4(8nq)^h$	$(n + 1)h + 3$	$j + 1$	$j + 1$	None	$j + 1$
Our	DBDH	No	$4(8nq)^h$	$h + n + 3$	$j + 1$	$j + 1$	None	$j + 1$

of the identity on which the operations are performed. The parameter h is the maximum number of levels in the HIBE. The construction in this paper requires $(h + n + 3)$ many elements of G_1 as public parameters whereas Waters HIBE requires $(n + 1)h + 3$ many elements. The security degradation remains the same in both cases.

5 Proof of Theorem 1

The security reduction follows along standard lines and develops on the proof given in [19,9,16]. We need to lower bound the probability of the simulator aborting on certain queries and in the challenge stage. The details of obtaining this lower bound are given in Section 5.1. In the following proof, we simply use the lower bound. We want to show that the HIBE is (ϵ_{hibe}, t, q) -CPA secure. In the game sequence style of proofs, we start with the adversarial game defining the CPA-security of the protocol against an adversary \mathcal{A} and then obtain a sequence of games as usual. In each of the games, the simulator chooses a bit δ and the adversary makes a guess δ' . By X_i we will denote the event that the bit δ is equal to the bit δ' in the i th game.

Game 0: This is the usual adversarial game used in defining CPA-secure HIBE. We assume that the adversary’s runtime is t and it makes q key extraction queries. Also, we assume that the adversary maximizes the advantage among all adversaries with similar resources. Thus, we have $\epsilon_{hibe} = |\Pr[X_0] - \frac{1}{2}|$.

Game 1: In this game, we setup the protocol from a tuple $\langle P, P_1 = aP, P_2 = bP, P_3 = cP, Z = e(P_1, P_2)^{abc} \rangle$ and answer key extraction queries and generate the challenge. The simulator is assumed to know the values a, b and c . However, the simulator can setup the protocol as well as answer certain private key queries without the knowledge of these values. Also, for certain challenge identities it can generate the challenge ciphertext without the knowledge of a, b and c . In the following, we show how this can be done. If the simulator cannot answer a key extraction query or generate a challenge without using the knowledge of a, b and c , it sets a flag flg to one. The value of flg is initially set to zero.

Note that the simulator is always able to answer the adversary (with or without using a, b and c). The adversary is provided with proper replies to all its queries and is also provided the proper challenge ciphertext. Thus, irrespective of whether flg is set to one, the adversary’s view in Game 1 is same as that in Game 0. Hence, we have $\Pr[X_0] = \Pr[X_1]$.

We next show how to setup the protocol and answer the queries based on the tuple $\langle P, P_1 = aP, P_2 = bP, P_3 = cP, Z = e(P_1, P_2)^{abc} \rangle$.

Set-Up: Let m be a prime such that $2q < m < 4q$. Our choice of m is different from that of previous works [19,9,16] where m was chosen to be equal to $4q$ and $2q$.

Choose x'_1, \dots, x'_h and x_1, \dots, x_n randomly from \mathbb{Z}_m ; also choose y'_1, \dots, y'_h and y_1, \dots, y_n randomly from \mathbb{Z}_p . Choose k_1, \dots, k_h randomly from $\{0, \dots, n\}$.

For $1 \leq j \leq h$, define $U'_j = (p - mk_j + x'_j)P_2 + y'_jP$ and for $1 \leq i \leq n$ define $U_i = x_iP_2 + y_iP$. The public parameters are $(P, P_1, P_2, U'_1, \dots, U'_h, U_1, \dots, U_n)$. The master secret is $aP_2 = abP$. The distribution of the public parameters is as expected by \mathcal{A} . In its attack, \mathcal{A} will make some queries, which have to be properly answered by the simulator.

For $1 \leq j \leq h$, we define several functions. Let $v = (v_1, \dots, v_n)$ where each $v_i \in \{0, 1\}$. We define

$$\left. \begin{aligned} F_j(v) &= p - mk_j + x'_j + \sum_{i=1}^n x_i v_i \\ J_j(v) &= y'_j + \sum_{i=1}^n y_i v_i \\ L_j(v) &= x'_j + \sum_{i=1}^n x_i v_i \pmod{m} \\ K_j(v) &= \begin{cases} 0 & \text{if } L_j(v) = 0 \\ 1 & \text{otherwise.} \end{cases} \end{aligned} \right\} \quad (2)$$

Recall that we have assumed $4q(n+1) < p$. Let F_{\min} and F_{\max} be the minimum and maximum values of $F_j(v)$. F_{\min} is achieved when k_j is maximum and x'_j and the x_i 's are all zero. Thus, $F_{\min} = p - mn$. We have $mn < 4q(n+1)$ and by assumption $4q(n+1) < p$. Hence, $F_{\min} > 0$. Again F_{\max} is achieved when $k_j = 0$ and x'_j and the x_i 's and v_i 's are equal to their respective maximum values. We get $F_{\max} < p + m(n+1) < p + 4q(n+1) < 2p$. Thus, we have $0 < F_{\min} \leq F_j(v) \leq F_{\max} < 2p$. Consequently, $F_j(v) \equiv 0 \pmod{p}$ if and only if $F_j(v) = p$ which holds if and only if $-mk_j + x'_j + \sum_{i=1}^n x_i v_i = 0$.

Now we describe how the queries made by \mathcal{A} are answered by \mathcal{B} . The queries can be made in both Phases 1 and 2 of the adversarial game (subject to the usual restrictions). The manner in which they are answered by the simulator is the same in both the phases.

Key Extraction Query: Suppose \mathcal{A} makes a key extraction query on the identity $\mathbf{v} = (v_1, \dots, v_j)$. Suppose there is a u with $1 \leq u \leq j$ such that $K_u(\mathbf{v}_u) = 1$. Otherwise set flg to one. In the second case, the simulator uses the value of a to return a proper private key $d_{\mathbf{v}} = (aP_2 + \sum_{i=1}^j r_i V_i, r_1 V_1, \dots, r_j V_j)$. In the first case, the simulator constructs a private key in the following manner.

Choose random r_1, \dots, r_j from \mathbb{Z}_p and define

$$\left. \begin{aligned} d_{0|u} &= -\frac{J_u(\mathbf{v}_u)}{F_u(\mathbf{v}_u)}P_1 + r_u(F_u(\mathbf{v}_u)P_2 + J_u(\mathbf{v}_u)P) \\ d_u &= \frac{-1}{F_u(\mathbf{v}_u)}P_1 + r_uP \\ d_k &= r_kP \text{ for } k \neq u \\ d_{\mathbf{v}} &= (d_{0|u} + \sum_{k \in \{1, \dots, j\} \setminus \{u\}} r_k V_k, d_1, \dots, d_j) \end{aligned} \right\} \quad (3)$$

The quantity d_v is a proper private key corresponding to the identity v . The algebraic verification of this fact is similar to that in [2,19]. This key is provided to \mathcal{A} .

Challenge: Let the challenge identity be $v^* = (v_1^*, \dots, v_{h^*}^*)$, $1 \leq h^* \leq h$ and the messages be M_0 and M_1 . Choose a random bit δ . We need to have $F_k(v_k^*) \equiv 0 \pmod p$ for all $1 \leq k \leq h^*$. If this condition does not hold, then set flg to one. In the second case, the simulator uses the value of c to provide a proper encryption of M_δ to \mathcal{A} by computing $(M_\delta \times e(P_1, P_2)^c, cP, cV_1, \dots, cV_{h^*})$. In the first case, it constructs a proper encryption of M_δ in the following manner.

$$(M^\delta \times Z, C_1 = P_3, B_1 = J_1(v_1^*)P_3, \dots, B_{h^*} = J_{h^*}(v_{h^*}^*)P_3).$$

We require B_j to be equal to $cV_j(v_j^*)$ for $1 \leq j \leq h^*$. Recall that the definition of $V_j(v)$ is $V_j(v) = U'_j + \sum_{k=1}^n v_k U_k$. Using the definition of U'_j and the U_k 's as defined in the setup by the simulator, we obtain, $cV_i(v_i^*) = c(F_i(v_i^*)P_2 + J_i(v_i^*)P) = J_i(v_i^*)cP = J_i(v_i^*)P_3$. Here we use the fact, $F_i(v_i^*) \equiv 0 \pmod p$. Hence, the quantities B_1, \dots, B_{h^*} are properly formed.

Guess: The adversary outputs a guess δ' of δ .

Game 2: This is a modification of Game 1 whereby the Z in Game 1 is now chosen to be a random element of G_2 . This Z is used to mask the message M_δ in the challenge ciphertext. Since Z is random, the first component of the challenge ciphertext is a random element of G_2 and provides no information to the adversary about δ . Thus, $\Pr[X_2] = \frac{1}{2}$.

We have the following claim.

Claim:

$$|\Pr[X_1] - \Pr[X_2]| \leq \frac{\epsilon_{dbdh}}{\lambda} + \frac{\epsilon_{hibe}}{2}.$$

Proof: The change from Game 1 to Game 2 corresponds to an “indistinguishability” step in Shoup’s tutorial [18] on such games. Usually, it is easy to bound the probability difference. In this case, the situation is complicated by the fact that there is a need to abort.

We show that it is possible to obtain an algorithm \mathcal{B} for DBDH by extending Games 1 and 2. The extension of both the games is same and is described as follows. \mathcal{B} takes as input a tuple (P, aP, bP, cP, Z) and sets up the HIBE protocol as in Game 1 (The setup of Games 1 and 2 are the same). The key extraction queries are answered and the challenge ciphertext is generated as in Game 1. If at any point of time flg is set to one by the game, then \mathcal{B} outputs a random bit and aborts. This is because the query cannot be answered or the challenge ciphertext cannot be generated using the input tuple. At the end of the game, the adversary outputs the guess δ' . \mathcal{B} now goes through a separate abort stage as follows.

“Artificial Abort”: The probability that \mathcal{B} aborts in the query or challenge phases depends on the adversary’s input. The goal of the artificial abort step is to make the probability of abort independent of the adversary’s queries by ensuring that in all cases its probability of abort is the maximum possible. This is done by sampling the transcript of adversary’s query and in certain cases aborting. The sampling procedure introduces the extra component $O(\epsilon_{hibe}^{-2} \ln(\epsilon_{hibe}^{-1}) \lambda^{-1} \ln(\lambda^{-1}))$ into the simulator’s runtime. (For details see [19,16].) Here λ is a lower bound on the probability that \mathcal{B} does not abort before entering the artificial abort stage. The expression for λ is obtained in Proposition 3 of Section 5.1.

Output: If \mathcal{B} has not aborted up to this stage, then it outputs 1 if $\delta = \delta'$; else 0.

Note that if Z is real, then the adversary is playing Game 1 and if Z is random, then the adversary is playing Game 2. The time taken by the simulator in either Game 1 or 2 is clearly $t + \chi(\epsilon_{hibe})$. From this point, standard inequalities and probability calculations establish the claim. \square

Now we can complete the proof in the following manner.

$$\begin{aligned} \epsilon_{hibe} &= \left| \Pr[X_0] - \frac{1}{2} \right| \\ &\leq |\Pr[X_0] - \Pr[X_2]| \\ &\leq |\Pr[X_0] - \Pr[X_1]| + |\Pr[X_1] - \Pr[X_2]| \\ &\leq \frac{\epsilon_{hibe}}{2} + \frac{\epsilon_{dbdh}}{\lambda}. \end{aligned}$$

Rearranging the inequality gives the desired result. This completes the proof of Theorem 1. \square

5.1 Lower Bound on Not Abort

We require the following two independence results in obtaining the required lower bound. Similar independence results have been used in [19,9,16] in connection with IBE protocols. The situation for HIBE is more complicated than IBE and especially so since we reuse some of the public parameters over different levels of the HIBE. This makes the proofs more difficult. Our independence results are given in Proposition 1 and 2 and these subsume the results of previous work. We provide complete proofs for these two propositions as well as a complete proof for the lower bound. The probability calculation for the lower bound is also more complicated compared to the IBE case.

Proposition 1. *Let m be a prime and $L(\cdot)$ be as defined in (2). Let $\mathbf{v}_1, \dots, \mathbf{v}_j$ be identities, i.e., each $\mathbf{v}_i = (v_{i,1}, \dots, v_{i,n})$, is an n -bit string. Then*

$$\Pr \left[\bigwedge_{k=1}^j (L_k(\mathbf{v}_k) = 0) \right] = \frac{1}{m^j}.$$

The probability is over independent and uniform random choices of $x'_1, \dots, x'_j, x_1, \dots, x_n$ from \mathbb{Z}_m . Consequently, for any $\theta \in \{1, \dots, j\}$, we have

$$\Pr \left[L_\theta(\mathbf{v}_\theta) = 0 \mid \bigwedge_{k=1, k \neq \theta}^j (L_k(\mathbf{v}_k) = 0) \right] = \frac{1}{m}.$$

Proof: Since \mathbb{Z}_m forms a field, we can do linear algebra with vector spaces over \mathbb{Z}_m . The condition $\bigwedge_{k=1}^j (L_j(\mathbf{v}_j) = 0)$ is equivalent to the following system of equations over \mathbb{Z}_m .

$$\begin{aligned} x'_1 + \mathbf{v}_{1,1}x_1 + \dots + \mathbf{v}_{1,n}x_n &= 0 \\ x'_2 + \mathbf{v}_{2,1}x_1 + \dots + \mathbf{v}_{2,n}x_n &= 0 \\ \dots &\dots \dots \dots \dots \dots \dots \\ x'_j + \mathbf{v}_{j,1}x_1 + \dots + \mathbf{v}_{j,n}x_n &= 0 \end{aligned}$$

This can be rewritten as

$$(x'_1, \dots, x'_j, x_1, \dots, x_n)A_{(j+n) \times (j+n)} = (0, \dots, 0)_{1 \times (j+n)}$$

where

$$A = \begin{bmatrix} I_j & O_{j \times n} \\ \mathbf{V}_{n \times j} & O_{n \times n} \end{bmatrix} \text{ and } \mathbf{V}_{n \times j} = \begin{bmatrix} \mathbf{v}_{1,1} & \dots & \mathbf{v}_{j,1} \\ \dots & \dots & \dots \\ \mathbf{v}_{1,n} & \dots & \mathbf{v}_{j,n} \end{bmatrix};$$

I_j is the identity matrix of order j ; O is the all zero matrix of the specified order. The rank of A is clearly j and hence the dimension of the solution space is n . Hence, there are m^n solutions in $(x'_1, \dots, x'_j, x_1, \dots, x_n)$ to the above system of linear equations. Since the variables $x'_1, \dots, x'_j, x_1, \dots, x_n$ are chosen independently and uniformly at random, the probability that the system of linear equations is satisfied for a particular choice of these variables is $m^n/m^{n+j} = 1/m^j$. This proves the first part of the result.

For the second part, note that we may assume $\theta = j$ by renaming the x' 's if required. Then

$$\Pr \left[L_j(\mathbf{v}_j) = 0 \mid \bigwedge_{k=1}^{j-1} (L_k(\mathbf{v}_k) = 0) \right] = \frac{\Pr \left[\bigwedge_{k=1}^j (L_k(\mathbf{v}_k) = 0) \right]}{\Pr \left[\bigwedge_{k=1}^{j-1} (L_k(\mathbf{v}_k) = 0) \right]} = \frac{m^{j-1}}{m^j} = \frac{1}{m}.$$

Proposition 2. Let m be a prime and $L(\cdot)$ be as defined in (2). Let $\mathbf{v}_1, \dots, \mathbf{v}_j$ be identities, i.e., each $\mathbf{v}_i = (\mathbf{v}_{i,1}, \dots, \mathbf{v}_{i,n})$, is an n -bit string. Let $\theta \in \{1, \dots, j\}$ and let \mathbf{v}'_θ be an identity such that $\mathbf{v}'_\theta \neq \mathbf{v}_\theta$. Then

$$\Pr \left[(L_\theta(\mathbf{v}'_\theta) = 0) \wedge \bigwedge_{k=1}^j (L_k(\mathbf{v}_k) = 0) \right] = \frac{1}{m^{j+1}}.$$

The probability is over independent and uniform random choices of $x'_1, \dots, x'_j, x_1, \dots, x_n$ from \mathbb{Z}_m . Consequently, we have

$$\Pr \left[L_\theta(\mathbf{v}'_\theta) = 0 \mid \bigwedge_{k=1}^j (L_k(\mathbf{v}_k) = 0) \right] = \frac{1}{m}.$$

Proof: The proof is similar to the proof of Proposition 1. Without loss of generality, we may assume that $\theta = j$, since otherwise we may rename variables to achieve this. The condition $(L_\theta(\mathbf{v}'_\theta) = 0) \wedge \bigwedge_{k=1}^j (L_k(\mathbf{v}_k) = 0)$ is equivalent to a system of linear equations $xA = 0$ over \mathbb{Z}_m . In this case, the form of A is the following.

$$A = \begin{bmatrix} I_j & c^T & O_{j \times n} \\ V_{n \times j} & (\mathbf{v}'_j)^T & O_{n \times n} \end{bmatrix}$$

where $c = (0, \dots, 0, 1)$; c^T denotes the transpose of c and $(\mathbf{v}'_j)^T$ is the transpose of \mathbf{v}'_j . The first j columns of A are linearly independent. The $(j + 1)$ th column of A is clearly linearly independent of the first $(j - 1)$ columns. We have $\mathbf{v}_j \neq \mathbf{v}'_j$ and $m > 2$, hence $\mathbf{v}_j \not\equiv \mathbf{v}'_j \pmod{m}$. Using this, it is not difficult to see that the first $(j + 1)$ columns of A are linearly independent and hence the rank of A is $(j + 1)$. Consequently, the dimension of the solution space is $n - 1$ and there are m^{n-1} solutions in $(x'_1, \dots, x'_j, x_1, \dots, x_n)$ to the system of linear equations. Since the x' 's and the x 's are chosen independently and uniformly at random from \mathbb{Z}_m , the probability of getting a solution is $m^{n-1}/m^{n+j} = 1/m^{j+1}$. This proves the first part of the result. The proof of the second part is similar to that of Proposition 1. \square

Proposition 3. *The probability that the simulator in the proof of Theorem 1 does not abort before the artificial abort stage is at least $\lambda = \frac{1}{2(4q(n+1))^h}$.*

Proof: We consider the simulator in the proof of Theorem 1. Up to the artificial abort stage, the simulator could abort on either a key extraction query or in the challenge stage. Let **abort** be the event that the simulator aborts before the artificial abort stage. For $1 \leq i \leq q$, let E_i denote the event that the simulator does not abort on the i th key extraction query and let C be the event that the simulator does not abort in the challenge stage. We have

$$\begin{aligned} \Pr[\overline{\text{abort}}] &= \Pr \left[\left(\bigwedge_{i=1}^q E_i \right) \wedge C \right] \\ &= \Pr \left[\left(\bigwedge_{i=1}^q E_i \right) \mid C \right] \Pr[C] \\ &= \left(1 - \Pr \left[\left(\bigvee_{i=1}^q \neg E_i \right) \mid C \right] \right) \Pr[C] \\ &\geq \left(1 - \sum_{i=1}^q \Pr[\neg E_i \mid C] \right) \Pr[C]. \end{aligned}$$

We first consider the event C . Let the challenge identity be $\mathbf{v}^* = (\mathbf{v}_1^*, \dots, \mathbf{v}_{h^*}^*)$. Event C holds if and only if $F_j(\mathbf{v}_j^*) \equiv 0 \pmod{p}$ for $1 \leq j \leq h^*$. Recall that by choice of p , we can assume $F_j(\mathbf{v}_j^*) \equiv 0 \pmod{p}$ if and only if $x'_j + \sum_{k=1}^n x_k \mathbf{v}_{j,k} = mk_j$. Hence,

$$\Pr[C] = \Pr \left[\bigwedge_{j=1}^{h^*} \left(x'_j + \sum_{k=1}^n x_k \mathbf{v}_{j,k} = m k_j \right) \right]. \tag{4}$$

For $1 \leq j \leq h^*$ and $0 \leq i \leq n$, denote the event $x'_j + \sum_{k=1}^n x_k \mathbf{v}_{j,k} = m_i$ by $A_{j,i}$ and the event $k_j = i$ by $B_{j,i}$. Also, let $C_{j,i}$ be the event $A_{j,i} \wedge B_{j,i}$.

Note that the event $\bigvee_{i=0}^n A_{j,i}$ is equivalent to $x'_j + \sum_{k=1}^n x_k \mathbf{v}_{j,k} \equiv 0 \pmod m$ and hence equivalent to the condition $L_j(\mathbf{v}_j) = 0$. Since k_j is chosen uniformly at random from the set $\{0, \dots, n\}$, we have $\Pr[B_{j,i}] = 1/(1+n)$ for all j and i . The events $B_{j,i}$'s are independent of each other and also independent of the $A_{j,i}$'s. We have

$$\begin{aligned} \Pr \left[\bigwedge_{j=1}^{h^*} \left(x'_j + \sum_{k=1}^n x_k \mathbf{v}_{j,k} = m k_j \right) \right] &= \Pr \left[\bigwedge_{j=1}^{h^*} \left(\bigvee_{i=0}^n C_{j,i} \right) \right] \\ &= \frac{1}{(1+n)^{h^*}} \Pr \left[\bigwedge_{j=1}^{h^*} \left(\bigvee_{i=0}^n A_{j,i} \right) \right] \\ &= \frac{1}{(1+n)^{h^*}} \Pr \left[\bigwedge_{j=1}^{h^*} (L_j(\mathbf{v}_j) = 0) \right] \\ &= \frac{1}{(m(1+n))^{h^*}} \end{aligned}$$

The last equality follows from Proposition 1.

Now we turn to bounding $\Pr[\neg E_i | C]$. For simplicity of notation, we will drop the subscript i from E_i and consider the event E that the simulator does not abort on a particular key extraction query on an identity $(\mathbf{v}_1, \dots, \mathbf{v}_j)$. By the simulation, the event $\neg E$ implies that $L_i(\mathbf{v}_i) = 0$ for all $1 \leq i \leq j$. This holds even when the event is conditioned under C . Thus, we have $\Pr[\neg E | C] \leq \Pr[\bigwedge_{i=1}^j L_i(\mathbf{v}_i) = 0 | C]$. The number of components in the challenge identity is h^* and now two cases can happen:

$j \leq h^*$: By the protocol constraint (a prefix of the challenge identity cannot be queried to the key extraction oracle), we must have a θ with $1 \leq \theta \leq j$ such that $\mathbf{v}_\theta \neq \mathbf{v}_\theta^*$.

$j > h^*$: In this case, we choose $\theta = h^* + 1$.

$$\Pr[\neg E | C] \leq \Pr \left[\bigwedge_{i=1}^j L_i(\mathbf{v}_i) = 0 | C \right] \leq \Pr[L_\theta(\mathbf{v}_\theta) = 0 | C] = 1/m.$$

The last equality follows from an application of either Proposition 1 or Proposition 2 according as whether $j > h^*$ or $j \leq h^*$. Substituting this in the bound for $\Pr[\overline{\text{abort}}]$ we obtain

$$\Pr[\overline{\text{abort}}] \geq \left(1 - \sum_{i=1}^q \Pr[\neg E_i | C] \right) \Pr[C].$$

$$\geq \left(1 - \frac{q}{m}\right) \frac{1}{(m(n+1))^h} \geq \frac{1}{2} \times \frac{1}{(4q(n+1))^h}.$$

We use $h \geq h^*$ and $2q < m < 4q$ to obtain the inequalities. This completes the proof. \square

6 Conclusion

Waters presented a construction of IBE [19] which significantly improves upon the previous construction of Boneh-Boyen [3]. In his paper, Waters also described a method to extend his IBE to a HIBE. The problem with this construction is that it increases the number public parameters. In this paper, we have presented a construction of a HIBE which builds upon the previous (H)IBE protocols. The number of public parameters is significantly less compared to Waters' HIBE. The main open problem in the construction of HIBE protocols is to avoid or control the security degradation which is exponential in the number of levels of the HIBE.

References

1. Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002.
2. Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Cachin and Camenisch [7], pages 223–238.
3. Dan Boneh and Xavier Boyen. Secure Identity Based Encryption Without Random Oracles. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, 2004.
4. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Cramer [10], pages 440–456. Full version available at Cryptology ePrint Archive; Report 2005/015.
5. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. Earlier version appeared in the proceedings of CRYPTO 2001.
6. Dan Boneh and Jonathan Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.
7. Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EURO-CRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.
8. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In Cachin and Camenisch [7], pages 207–222.
9. Sanjit Chatterjee and Palash Sarkar. Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model. In Dong Ho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 424–440. Springer, 2005.

10. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.
11. Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the Tate Pairing. In Claus Fieker and David R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.
12. David Galindo. Boneh-Franklin Identity Based Encryption Revisited. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 791–802. Springer, 2005.
13. Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.
14. Jeremy Horwitz and Ben Lynn. Toward Hierarchical Identity-Based Encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.
15. Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004. Earlier version appeared in the proceedings of ANTS-IV.
16. David Naccache. Secure and Practical Identity-Based Encryption. Cryptology ePrint Archive, Report 2005/369, 2005. <http://eprint.iacr.org/>.
17. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.
18. Victor Shoup. Sequences of Games: a Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2004. <http://eprint.iacr.org/>.
19. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [10], pages 114–127.

Forward-Secure and Searchable Broadcast Encryption with Short Ciphertexts and Private Keys

Nuttapong Attrapadung¹, Jun Furukawa², and Hideki Imai³

¹ Institute of Industrial Science, University of Tokyo, Japan
nuts@imailab.iis.u-tokyo.ac.jp

² NEC Corporation, Japan
j-furukawa@ay.jp.nec.com

³ Research Center for Information Security, AIST, Japan
h-imai@aist.go.jp

Abstract. We introduce a primitive called *Hierarchical Identity-Coupling Broadcast Encryption* (HICBE) that can be used for constructing efficient collusion-resistant public-key broadcast encryption schemes with extended properties such as forward-security and keyword-searchability. Our forward-secure broadcast encryption schemes have small ciphertext and private key sizes, in particular, independent of the number of users in the system. One of our best two constructions achieves ciphertexts of constant size and user private keys of size $O(\log^2 T)$, where T is the total number of time periods, while another achieves both ciphertexts and user private keys of size $O(\log T)$. These performances are comparable to those of the currently best *single-user* forward-secure public-key encryption scheme, while our schemes are designed for broadcasting to arbitrary sets of users. As a side result, we also formalize the notion of searchable broadcast encryption, which is a new generalization of public key encryption with keyword search. We then relate it to anonymous HICBE and present a construction with polylogarithmic performance.

1 Introduction

Broadcast encryption (BE) scheme [16] allows a broadcaster to encrypt a message to an arbitrarily designated subset S of all users in the system. Any user in S can decrypt the message by using his own private key while users outside S should not be able to do so even if all of them collude. Such a scheme is motivated by many applications such as pay-TV systems, the distribution of copyrighted materials such as CD/DVD. Public-key broadcast encryption is the one in which the broadcaster key is public. Such a scheme is typically harder to construct than private-key type ones. In what follows, we let n denote the number of all users.

The best BE scheme so far in the literature was recently proposed by Boneh, Gentry, and Waters [7]. Their scheme, which is a public-key scheme, achieves asymptotically optimal sizes, $O(1)$, for both broadcast ciphertexts and user private keys, with the price of $O(n)$ -size public key. (To achieve some tradeoff, they

also proposed a generalized scheme, of which one parametrization gives a scheme where both the public keys and the ciphertexts are of size $O(\sqrt{n})$. The previously best schemes [20,19,18], along the line of the subset-cover paradigm by Naor, Naor, and Lotspiech (NNL) [20], can only achieve a broadcast ciphertext of size $O(r)$ with each user's private key being of size $O(\log n)$, where $r = n - |S|$ is the number of revoked users. Although these schemes are improved in [3] by reducing the private key size to $O(1)$, the ciphertext is still of size $O(r)$.¹ These NNL derivatives are originally private-key schemes. Dodis and Fazio [15] gave a framework to extend these schemes to public-key versions using Hierarchical Identity-Based Encryption (HIBE) [17]. Instantiating this framework with a recent efficient HIBE scheme by Boneh, Boyen, and Goh [5] gives a public-key version of NNL-based schemes without loss in performance of ciphertext sizes.

Forward-Secure Broadcast Encryption. Unfortunately, a normal broadcast encryption scheme offers no security protection for any user whatsoever once his private key is compromised. As an extension to the normal variant in order to cope with the vulnerability against key exposure, the notion of forward security in the context of public-key broadcast encryption was first studied by Yao et al. [22]. A forward-secure public-key broadcast encryption (FS-BE) allows each user to update his private key periodically while keeping the public key unchanged. Such a scheme guarantees that even if an adversary learns the private key of some user at time period τ , messages encrypted during all time periods prior to τ remain secret. Yao et al. also proposed a FS-BE scheme achieving ciphertexts of size $O(r \log T \log n)$ while each user's private key is of size $O(\log^3 n \log T)$, where T is the maximum allowed time period. Indeed, they proposed a forward-secure HIBE scheme and then applied it to the NNL scheme in essentially the same manner as done by [15], as mentioned above. Later, Boneh et al. [5] proposed (at least two) more efficient forward-secure HIBE schemes, which when applying to the NNL scheme gives a FS-BE scheme with ciphertexts of size $O(r)$ and private keys of size $O(\log^3 n \log T)$ and another FS-BE scheme with ciphertexts of size $O(r \log T)$ and private keys of size $O((\log^2 n)(\log n + \log T))$. These schemes are the best FS-BE schemes so far in the literature.

1.1 Our Contributions

Towards constructing a more efficient FS-BE scheme, we introduce a new primitive called *Hierarchical Identity-Coupling Broadcast Encryption* (HICBE), which can be considered as a generalization either of BE that further includes hierarchical-identity dimension together with key derivation functionality or of HIBE that further includes a user dimension together with broadcast functionality. Besides forward security, HICBE can be used to construct BE with other extended properties such as keyword-searchability, which is another feature that we study as a side result in this paper (see below).

¹ Note that one advantage of these NNL-based schemes is that, in contrast to the BGW scheme, all the other efficiency parameters, beside ciphertext sizes and private key sizes, are also of sub-linear (in n) size.

FS-BE with Short Ciphertexts and Private Keys. Using HICBE as a building block, we propose at least three new FS-BE schemes. One of our best two schemes achieves ciphertexts of size $O(1)$ and user private keys of size $O(\log^2 T)$. The other best scheme achieves ciphertexts of size $O(\log T)$ and user private keys of size $O(\log T)$. These outperform the previous schemes in terms of both overheads. In particular, they are independent of the parameters in the user dimension, namely n and r ; moreover, the first scheme achieves the constant-size ciphertext. These performances of our schemes are comparable to those of the currently best single-user forward-secure public-key encryption scheme (cf. [5]). The public keys for both schemes are of size $O(n + \log T)$. Analogously to [7], we can show that this amount can be traded off to $O(\sqrt{n} + \log T)$ with ciphertext size being increased to $O(\sqrt{n})$ and $O(\sqrt{n} + \log T)$ respectively in both schemes.

Security of our systems is based on the Decision Bilinear Diffie-Hellman Exponent assumption (BDHE), which is previously used in [7,5]. We prove the security in the standard model (i.e., without random oracle).

Searchable Broadcast Encryption. Public-key BE can be applied naturally to encrypted file systems, which enable file sharing among privileged users over a public server, as already suggested in [7]. A file can be created by anyone using the public key and the privileged subset can be arbitrarily specified by the creator of the file. Due to a possible large amount of databases, a user Alice might want to retrieve only those files that contain a particular keyword of interest (among all the files in which Alice is specified as a privileged user), but without giving the server the ability to decrypt the databases. *Public-key Broadcast Encryption with Keyword Search* (BEKS) allows to do exactly this. It enables Alice to give the server a capability (or a trapdoor) to test whether a particular keyword, w , is contained in any (and only) file that includes Alice as a privileged user. This is done in such a way that (1) the server is unable to learn anything else about that file, besides the information about containment of w , and (2) all the other users outside the privileged set cannot learn anything, in particular, cannot generate such a trapdoor, even if they collude.

BEKS is a new generalization of public key encryption with keyword search (PEKS) [6] that we introduce in this paper. We then relate that an anonymous ICBE (1-level HICBE) is sufficient to construct BEKS, analogously to the relation between anonymous IBE and PEKS [1].

A trivial BEKS achieving ciphertexts of size $O(n)$ can be constructed from the concatenation of PEKS-encryption of the same keyword to each privileged user. Our scheme achieves ciphertexts of size $O(r \log n)$, trapdoors of size $O(\log^3 n)$, and private keys of size $O(\log^4 n)$. Before coming up with this result, we constructively hint that even using the same technique as our FS-BE schemes (where a *non-anonymous* HICBE is sufficient), it might not be easy to construct a BEKS scheme with both ciphertext and private key of sizes independent of n . We refer for most of the results in this part to the full version of this paper [2] due to limited space here.

2 Preliminaries

Bilinear Maps. We briefly review facts about bilinear maps. We use the standard terminology from [8]. Let \mathbb{G}, \mathbb{G}_1 be multiplicative groups of prime order p . Let g be a generator of \mathbb{G} . A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ for which the following hold: (1) e is bilinear; that is, for all $u, v \in \mathbb{G}$, $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$. (2) The map is non-degenerate: $e(g, g) \neq 1$. We say that \mathbb{G} is a bilinear group if the group action in \mathbb{G} can be computed efficiently and there exists \mathbb{G}_1 for which the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is efficiently computable. Although it is desirable to use asymmetric type, $e : \mathbb{G} \times \mathbb{G}' \rightarrow \mathbb{G}_1$ where $\mathbb{G} \neq \mathbb{G}'$, so that group elements will have compact representation, for simplicity we will present our schemes by the symmetric ones. Indeed, our schemes can be rephrased in terms of asymmetric maps.

Decision BDHE Assumption.² Let \mathbb{G} be a bilinear group of prime order p . The Decision n -BDHE (Bilinear Diffie-Hellman Exponent) problem [7,5] in \mathbb{G} is stated as follows: given a vector

$$\left(g, h, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^n}, g^{\alpha^{n+2}}, \dots, g^{\alpha^{2n}}, Z \right) \in \mathbb{G}^{2n+1} \times \mathbb{G}_1$$

as input, determine whether $Z = e(g, h)^{\alpha^{n+1}}$. We denote $g_i = g^{\alpha^i} \in \mathbb{G}$ for shorthand. Let $\mathbf{y}_{g,\alpha,n} = (g_1, \dots, g_n, g_{n+2}, \dots, g_{2n})$. An algorithm \mathcal{A} that outputs $b \in \{0, 1\}$ has advantage ϵ in solving Decision n -BDHE in \mathbb{G} if $|\Pr[\mathcal{A}(g, h, \mathbf{y}_{g,\alpha,n}, e(g_{n+1}, h)) = 0] - \Pr[\mathcal{A}(g, h, \mathbf{y}_{g,\alpha,n}, Z) = 0]| \geq \epsilon$, where the probability is over the random choice of generators $g, h \in \mathbb{G}$, the random choice of $\alpha \in \mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_1$, and the randomness of \mathcal{A} . We refer to the distribution on the left as \mathcal{P}_{BDHE} and the distribution on the right as \mathcal{R}_{BDHE} . We say that the Decision (t, ϵ, n) -BDHE assumption holds in \mathbb{G} if no t -time algorithm has advantage at least ϵ in solving the Decision n -BDHE problem in \mathbb{G} . We sometimes drop t, ϵ and refer it as the Decision n -BDHE assumption in \mathbb{G} .

3 Hierarchical Identity-Coupling Broadcast Encryption

Model. A HICBE system consists of n users, each with index $i \in \{1, \dots, n\}$. In usage, a user index will be “coupled” with some additional arbitrary identity tuple $\text{ID} = (I_1, \dots, I_z)$, for any I_j in some predefined identity space \mathcal{I} and any $z = 1, \dots, L$ where L is a predetermined maximum depth of tuples. The user i coupling with ID , which we will refer as a *node* (i, ID) , will possess its own private key $d_{i,\text{ID}}$. If $\text{ID} = (I_1, \dots, I_z)$, then for $j = 1, \dots, z$, let $\text{ID}|_j = (I_1, \dots, I_j)$, and let $\text{ID}|_0$ be the empty string ε . A HICBE system enables a derivation from

² This holds in the generic bilinear group model with the computational lower bound of $\Omega(\sqrt{p/n})$ on the difficulty of breaking (cf.[5]). Cheon [14] recently showed a concrete attack with roughly the same complexity. It is recommended to either increase p (to ≈ 220 -bit size for $n = 2^{64}$ to achieve 2^{80} security) or use p of a special form where $p - 1$ and $p + 1$ have no small divisor greater than $\log^2 p$ to avoid the attack.

$d_{i, \text{ID}_{|z-1}}$ to $d_{i, \text{ID}}$. In particular, $d_{i, (\text{I}_1)}$ can be derived from d_i , the *root* private keys of i . A HICBE system enables one to encrypt a message to a set of nodes $\{(i, \text{ID}) | i \in S\}$ for arbitrary $S \subseteq \{1, \dots, n\}$, where we say that it is encrypted to *multi-node* (S, ID) . If $i \in S$, the user i coupling with ID (who possesses $d_{i, \text{ID}}$) can decrypt this ciphertext. When $L = 1$, we simply call it an ICBE.

Formally, a HICBE system is made up of five randomized algorithms as follows. For simplicity, we define it as a key encapsulation mechanism (KEM).

Setup(n, L): Takes as input the number of all users n and the maximum depth L of the identity hierarchy. It outputs a public key pk and a master key msk .

PrivKeyGen(i, pk, msk): Takes as input a user index i , the public key pk , and the master key msk . It outputs a root private key d_i of user i .

Derive($\text{pk}, i, \text{ID}, d_{i, \text{ID}_{|z-1}}$): Takes as input the public key pk , a user index i , an identity ID of depth z , and the private key $d_{i, \text{ID}_{|z-1}}$ of user i coupling with the parent identity $\text{ID}_{|z-1}$. It outputs $d_{i, \text{ID}}$. Here $d_{i, \text{ID}_{|0}} = d_i$.

Encrypt(pk, S, ID): Takes as input the public key pk , a subset $S \subseteq \{1, \dots, n\}$, and an identity tuple ID . It outputs a pair (hdr, K) where hdr is called the header and $K \in \mathcal{K}$ is a message encryption key. We will also refer to hdr as the broadcast ciphertext.

Decrypt($\text{pk}, S, i, d_{i, \text{ID}}, \text{hdr}$): Takes as input the pk , a subset S , a user i , the private key $d_{i, \text{ID}}$ of user i coupling with ID , and the header hdr . If $i \in S$ it outputs $K \in \mathcal{K}$ else outputs a special symbol \nexists .

The correctness consistency can be defined straightforwardly and is omitted here.

Confidentiality. We define semantic security of HICBE by the following game between an adversary \mathcal{A} and a challenger \mathcal{C} ; both are given n, L as input.

Setup. The challenger \mathcal{C} runs **Setup**(n, L) to obtain a public key pk and the master key msk . It then gives the public key pk to \mathcal{A} .

Phase 1. \mathcal{A} adaptively issues queries q_1, \dots, q_μ where each is one of two types:

- Private key query $\langle i, \text{ID} \rangle$. \mathcal{C} responds by running algorithm **PrivKeyGen** and **Derive** to derive the private key $d_{i, \text{ID}}$, corresponding to the node (i, ID) , then sends $d_{i, \text{ID}}$ to \mathcal{A} .
- Decryption query $\langle S, \text{ID}, i, \text{hdr} \rangle$ where $i \in S$. \mathcal{C} responds by running algorithm **PrivKeyGen** and **Derive** to derive the private key $d_{i, \text{ID}}$, corresponding to the node (i, ID) . It then gives to \mathcal{A} the output from **Decrypt**($\text{pk}, S, i, d_{i, \text{ID}}, \text{hdr}$).

Challenge. Once \mathcal{A} decides that Phase 1 is over, it outputs (S^*, ID^*) which is the multi-node it wants to attack, where $S^* \subseteq \{1, \dots, n\}$. The only restriction is that \mathcal{A} did not previously issue a private key query for $\langle i, \text{ID} \rangle$ such that $i \in S^*$ and that either $\text{ID} = \text{ID}^*$ or ID is a prefix of ID^* . \mathcal{C} then compute $(\text{hdr}^*, K) \xleftarrow{R} \text{Encrypt}(\text{pk}, S^*, \text{ID}^*)$ where $K \in \mathcal{K}$. Next \mathcal{C} picks a random $b \in \{0, 1\}$. It sets $K_b = K$ and picks a random K_{1-b} in \mathcal{K} . It then gives (hdr^*, K_0, K_1) to \mathcal{A} .

Phase 2. \mathcal{A} issues additional queries $q_{\mu+1}, \dots, q_\nu$ where each is one of two types:

- Private key query $\langle i, \text{ID} \rangle$ such that if $i \in S^*$ then neither $\text{ID} = \text{ID}^*$ nor ID is a prefix of ID^* , else $(i \notin S^*)$ ID can be arbitrary.

- Decryption query $(S, \text{ID}, i, \text{hdr})$ where $i \in S$ and $S \subseteq S^*$.³ The only constraint is that $\text{hdr} \neq \text{hdr}^*$ if either $\text{ID} = \text{ID}^*$ or ID is a prefix of ID^* .

In both cases, \mathcal{C} responds as in Phase 1. These queries may be adaptive.

Guess. Finally \mathcal{A} outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

We refer to such an adversary \mathcal{A} as an **IND-aID-aSet-CCA** adversary and the above game as the **IND-aID-aSet-CCA** game. Weaker notions of security can be defined by modifying the above game so that it is required that the adversary must commit ahead of time to the target subset S^* or the target identity ID^* or both. These notions are analogous to the notion of selective-identity secure HIBE, defined in [12,13]. We have 4 possible combinations: the game **IND-xID-ySet-CCA** where $(x, y) \in \{(a, a), (a, s), (s, a), (s, s)\}$. If $(x, y) = (s, *)$ then it is exactly the same as **IND-aID-aSet-CCA** except that \mathcal{A} must disclose to \mathcal{C} the target identity ID^* before the Setup phase. Analogously, if $(x, y) = (*, s)$, \mathcal{A} must disclose the target subset S^* before the Setup phase. For only the case of (s, s) , it is further required that the restrictions on private key queries from phase 2 also hold in phase 1. Intuitively, **s** means selective while **a** means adaptive security.

We define the advantage of the adversary \mathcal{A} in attacking the HICBE scheme \mathcal{E} in the game **IND-xID-ySet-CCA** as $\text{AdvHICBE}_{xy}(\mathcal{E}, \mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$, where the probability is over the random bits used by \mathcal{C} and \mathcal{A} in that game.

Definition 1. We say that a HICBE system \mathcal{E} is (t, q_P, q_D, ϵ) -**IND-xID-ySet-CCA-secure** if for any t -time **IND-xID-ySet-CCA** adversary \mathcal{A} that makes at most q_P chosen private key queries and at most q_D chosen decryption queries, we have that $\text{AdvHICBE}_{xy}(\mathcal{E}, \mathcal{A}) < \epsilon$. We say that a HICBE system \mathcal{E} is (t, q_P, ϵ) -**IND-xID-ySet-CPA-secure** if \mathcal{E} is $(t, q_P, 0, \epsilon)$ -**IND-xID-ySet-CCA-secure**.

Anonymity. Recipient anonymity is the property that the adversary be unable to distinguish the ciphertext intended for a chosen identity from another one intended for a random identity. We capture such a property via what we name **ANO-xID-ySet-CCA[Δ]** notion, where $\Delta \subseteq \{0, \dots, L\}$ indicates a set of levels that satisfy anonymity, with 0 corresponds to the anonymity of the set S . This is a generalized notion from [1]. We refer to the full paper [2] for the details .

4 HICBE Constructions

In this section, we give our first two HICBE constructions. A HICBE system must have both broadcast and hierarchical-identity-based derivation properties. To achieve this we will combine some techniques from the BGW broadcast encryption [7] with the BB and BBG HIBE systems by Boneh-Boyen [4] and Boneh-Boyen-Goh [5] respectively. The reader is encouraged to refer to the full paper [2] for the intuition into the design.

³ It is WLOG that we just restrict $S \subseteq S^*$ since for S such that $S \not\subseteq S^*$, one can make a private key query for some $i \in S \setminus S^*$ and perform the decryption oneself.

4.1 Our First HICBE Construction Based on BGW and BB

We first show how to combine the basic BGW scheme with the BB HIBE scheme. We assume that the identity space \mathcal{I} is \mathbb{Z}_p . Thus, if ID is of depth z then $\text{ID} = (I_1, \dots, I_z) \in \mathbb{Z}_p^z$. As in [4], we can later extend the construction to arbitrary identities in $\{0, 1\}^*$ by first hashing each I_j using a collision resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. We follow almost the same terminology from [7,4]. This scheme, denoted by BasicHICBE1, works as follows.

Setup(n, L): Let \mathbb{G} be a bilinear group of prime order p . It first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i = 1, 2, \dots, n, n+2, \dots, 2n$. Next, it picks a random $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$. It then picks random elements $h_1, \dots, h_L \in \mathbb{G}$. The public key is:

$$\text{pk} = (g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, v, h_1, \dots, h_L) \in \mathbb{G}^{2n+L+1}.$$

The master key is $\text{msk} = \gamma$. For $j = 1, \dots, L$, we define $F_j : \mathbb{Z}_p \rightarrow \mathbb{G}$ to be the function: $F_j(x) = g_1^x h_j$. The algorithm outputs pk and msk .

PrivKeyGen(i, pk, msk): Set a root private key for i as $d_i = (g_i)^\gamma = v^{(\alpha^i)} \in \mathbb{G}$.

Derive($\text{pk}, i, \text{ID}, d_{i, \text{ID}|_{z-1}}$): To generate the private key for node (i, ID) where $i \in \{1, \dots, n\}$ and $\text{ID} = (I_1, \dots, I_z) \in \mathbb{Z}_p^z$ of depth $z \leq L$, pick random elements $s_1, \dots, s_z \in \mathbb{Z}_p$ and output

$$d_{i, \text{ID}} = \left((g_i)^\gamma \cdot \prod_{j=1}^z F_j(I_j)^{s_j}, g^{s_1}, \dots, g^{s_z} \right) \in \mathbb{G}^{z+1}.$$

Note that the private key for node (i, ID) can be generated just given a private key for node $(i, \text{ID}|_{z-1})$ where $\text{ID}|_{z-1} = (I_1, \dots, I_{z-1}) \in \mathbb{Z}_p^{z-1}$, as required. Indeed, let $d_{i, \text{ID}|_{z-1}} = (a_0, \dots, a_{z-1})$ be the private key for node $(i, \text{ID}|_{z-1})$. To generate $d_{i, \text{ID}}$, pick a random $s_z \in \mathbb{Z}_p$ and output $d_{i, \text{ID}} = (a_0 \cdot F_z(I_z)^{s_z}, a_1, \dots, a_{z-1}, g^{s_z})$.

Encrypt(pk, S, ID): Pick a random $t \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^t$. The value $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$. Let $\text{ID} = (I_1, \dots, I_z)$. It outputs (hdr, K) where we let

$$\text{hdr} = \left(g^t, \left(v \cdot \prod_{j \in S} g_{n+1-j} \right)^t, F_1(I_1)^t, \dots, F_z(I_z)^t \right) \in \mathbb{G}^{z+2}.$$

Decrypt($\text{pk}, S, i, d_{i, \text{ID}}, \text{hdr}$): Parse the header as $\text{hdr} = (C_0, C_1, A_1, \dots, A_z) \in \mathbb{G}^{z+2}$. Also parse $d_{i, \text{ID}} = (a_0, \dots, a_z) \in \mathbb{G}^{z+1}$. Then output

$$K = e(g_i, C_1) \cdot \prod_{j=1}^z e(A_j, a_j) / e(a_0 \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, C_0).$$

The correctness verification is straightforward. The scheme inherits a good property of the BGW scheme: the ciphertext size and user private key size are independent of n . Indeed, when we let $\text{ID} = \varepsilon$, the corresponding algorithms become those of the basic BGW scheme.

Theorem 1. *Let \mathbb{G} be a bilinear group of prime order p . Suppose the Decision (t, ϵ, n) -BDHE assumption holds in \mathbb{G} . Then the BasicHICBE1 system for n users and maximum depth L is (t', q_P, ϵ) -IND-sID-sSet-CPA-secure for any n, L, q_P , and $t' < t - \Theta(\tau_{\text{exp}} L q_P)$ where τ_{exp} is the maximum time for an exponentiation in \mathbb{G} .*

The security proof, although vaguely resembles those of BGW and BB, is not straightforward as we have to simulate both sub-systems simultaneously. Intuitively, the implicit “orthogonality” of BGW and BB allows us to prove the security of the combined scheme. We omit it here (and refer to [2]) and will focus on a similar but somewhat more interesting proof of the second scheme.

4.2 Our Second HICBE Construction Based on BGW and BBG

Our method of integrating the BGW system can also be applied to the BBG HIBE scheme analogously to the previous integration. In contrast, this time we achieve a feature of “reusing” the public key from the BGW portion to be used for the BBG portion. Consequently, the resulting scheme has exactly the same public key as the BGW scheme except for only one additional element of \mathbb{G} .

We will assume that $L \leq n$, otherwise just create dummy users so as to be so; a more efficient way will be discussed in the next subsection. As usual we can assume that \mathcal{I} is \mathbb{Z}_p . The scheme, denoted by BasicHICBE2, works as follows.

Setup(n, L): The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i = 1, 2, \dots, n, n + 2, \dots, 2n$. Next, it randomly picks $y \in \mathbb{G}$, $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$. The public key is:

$$\text{pk} = (g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, v, y) \in \mathbb{G}^{2n+2}.$$

The master key is $\text{msk} = \gamma$. It outputs (pk, msk) . For conceptual purpose, let $h_j = g_{n+1-j}$ for $j = 1, \dots, L$; intuitively, the h_j terms will be used to visually indicate the BBG portion, while the g_j terms are for the BGW portion.

PrivKeyGen(i, pk, msk): Set a root private key for i as $d_i = (g_i)^\gamma = v^{(\alpha^i)} \in \mathbb{G}$.

Derive($\text{pk}, i, \text{ID}, d_{i, \text{ID}}|_{z-1}$): To generate the private key for node (i, ID) where $i \in \{1, \dots, n\}$ and $\text{ID} = (I_1, \dots, I_z) \in \mathbb{Z}_p^z$ of depth $z \leq L$, pick a random element $s \in \mathbb{Z}_p$ and output

$$d_{i, \text{ID}} = \left((g_i)^\gamma \cdot (h_1^{I_1} \cdots h_z^{I_z} \cdot y)^s, g^s, h_{z+1}^s, \dots, h_L^s \right) \in \mathbb{G}^{2+L-z}.$$

Note that the private key for node (i, ID) can be generated just given a private key for node $(i, \text{ID}|_{z-1})$ where $\text{ID}|_{z-1} = (I_1, \dots, I_{z-1}) \in \mathbb{Z}_p^{z-1}$, as required. Indeed, let $d_{i, \text{ID}|_{z-1}} = (a_0, a_1, b_z, \dots, b_L)$ be the private key for node $(i, \text{ID}|_{z-1})$. To generate $d_{i, \text{ID}}$, pick a random $\delta \in \mathbb{Z}_p$ and output $d_{i, \text{ID}} = \left(a_0 \cdot b_z^{I_z} \cdot (h_1^{I_1} \cdots h_z^{I_z} \cdot y)^\delta, a_1 \cdot g^\delta, b_{z+1} \cdot h_{z+1}^\delta, \dots, b_L \cdot h_L^\delta \right)$. This key has a proper distribution as a private key for node (i, ID) with the randomness $s = s' + \delta \in \mathbb{Z}_p$, where s' is the randomness in $d_{i, \text{ID}|_{z-1}}$. Note that the private key $d_{i, \text{ID}}$ becomes shorter as the depth of ID increases.

Encrypt(pk, S, ID): Pick a random $t \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^t$. The value $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$. Let $ID = (I_1, \dots, I_z)$. It outputs (hdr, K) where we let

$$\text{hdr} = \left(g^t, \left(v \cdot \prod_{j \in S} g_{n+1-j} \right)^t, \left(h_1^{I_1} \cdots h_z^{I_z} \cdot y \right)^t \right) \in \mathbb{G}^3.$$

Decrypt(pk, S, i, $d_{i, ID}$, hdr): Let $\text{hdr} = (C_0, C_1, C_2) \in \mathbb{G}^3$ and let $d_{i, ID} = (a_0, a_1, b_{z+1}, \dots, b_L) \in \mathbb{G}^{2+L-z}$. Then output

$$K = e(g_i, C_1) \cdot e(C_2, a_1) / e(a_0 \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, C_0).$$

The scheme inherits good properties from both the BGW scheme: the ciphertext size and user private key size are independent of n , and the BBG scheme: the ciphertext size is constant. One difference from the BBG system is that we let the h_j terms be of special forms, namely $h_j = g_{n+1-j}$, instead of random elements in \mathbb{G} as in [5]. This allows us to save the public key size since those g_j terms are already used for the BGW system. Indeed, suppose that the BGW BE system has been already established, it can be augmented to a HICBE version by just once publishing one random element, namely $y \in \mathbb{G}$, as an additional public key. Note that defining h_j terms in this way is also crucial to the security proof. We prove the security under the Decision n -BDHE assumption. This strong assumption is already necessary for both the (stand-alone) BGW and BBG systems.⁴

Theorem 2. *Let \mathbb{G} be a bilinear group of prime order p . Suppose the Decision (t, ϵ, n) -BDHE assumption holds in \mathbb{G} . Then the BasicHICBE2 scheme for n users and maximum depth L is (t', q_p, ϵ) -IND-sID-sSet-CPA-secure for arbitrary n, L such that $L \leq n$ and q_p , and any $t' < t - \Theta(\tau_{\text{exp}} L q_p)$ where τ_{exp} is the maximum time for an exponentiation in \mathbb{G} .*

Proof. Suppose there exists an adversary, \mathcal{A} , that has advantage ϵ in attacking the HICBE scheme. We build an algorithm \mathcal{B} that solves the Decision n -BDHE problem in \mathbb{G} . \mathcal{B} is given as input a random n -BDHE challenge $(g, h, \mathbf{y}_{g, \alpha, n}, Z)$, where $\mathbf{y}_{g, \alpha, n} = (g_1, \dots, g_n, g_{n+2}, \dots, g_{2n})$ and Z is either $e(g_{n+1}, h)$ or a random element in \mathbb{G}_1 (recall that $g_j = g^{(\alpha^j)}$). Algorithm \mathcal{B} proceeds as follows.

Initialization. The selective (identity, subset) game begins with \mathcal{A} first outputting a multi-node (S^*, ID^*) where $S^* \subseteq \{1, \dots, n\}$ and $ID^* = (I_1^*, \dots, I_z^*) \in \mathbb{Z}_p^z$ of depth $z \leq L$ that it intends to attack.

Setup. To generate pk, algorithm \mathcal{B} randomly chooses $u, \sigma \in \mathbb{Z}_p$ and sets

$$v = g^u \cdot \left(\prod_{j \in S^*} g_{n+1-j} \right)^{-1}, \quad y = g^\sigma \cdot \prod_{j=1}^z g_{n+1-j}^{-I_j^*}.$$

⁴ It was later shown in [5, full] that a *truncated* form of Decision n -BDHE, namely the Decision n -wBDHI*, indeed suffices for BBG. This assumption is defined exactly the same as the former except that we change the vector $\mathbf{y}_{g, \alpha, n}$ to $\mathbf{y}_{g, \alpha, n}^* := (g_1, \dots, g_n)$.

It gives \mathcal{A} the $\text{pk} = (g, \mathbf{y}_{g,\alpha,n}, v, y)$. Since g, α, u, σ are chosen randomly and independently, pk has an identical distribution to that in the actual construction.

Phase 1. \mathcal{A} issues up to q_P private key queries. Consider a query for the private key corresponding to node (i, ID) , of which $\text{ID} = (I_1, \dots, I_w) \in \mathbb{Z}_p^w$ where $w \leq L$. We distinguish two cases according to whether i is in S^* or not.

If $i \notin S^*$ then \mathcal{B} responds to the query by first computing a root private key d_i from which it can then construct a private key $d_{i,\text{ID}}$ for the request node (i, ID) . In this case, \mathcal{B} computes d_i as $d_i = g_i^u \cdot (\prod_{j \in S^*} g_{n+1-j+i})^{-1}$. Indeed, we have $d_i = (g^u (\prod_{j \in S^*} g_{n+1-j})^{-1})^{\alpha^i} = v^{\alpha^i}$, as required.

If $i \in S^*$ then from the restriction of the private key query, it must be that ID is neither ID^* nor any prefix of ID^* . We further distinguish two cases according to whether ID^* is a prefix of ID or not.

Case 1: ID^* is *not* a prefix of ID . Then there must exist $k \leq z$ such that it is the smallest index satisfying $I_k \neq I_k^*$. \mathcal{B} responds to the query by first computing a private key for node $(i, \text{ID}_{|k})$ from which it then constructs a private key for the request node (i, ID) . \mathcal{B} picks random elements $s \in \mathbb{Z}_p$. We pose $\tilde{s} = s + \alpha^k / (I_k - I_k^*)$. Note that \tilde{s} is unknown to \mathcal{B} . Next, \mathcal{B} generates the private key

$$(a_0, a_1, b_{k+1}, \dots, b_L) = \left(v^{\alpha^i} \cdot (h_1^{I_1} \dots h_k^{I_k} \cdot y)^{\tilde{s}}, g^{\tilde{s}}, h_{k+1}^{\tilde{s}}, \dots, h_L^{\tilde{s}} \right) \quad (1)$$

which is a valid random private key for node $(i, \text{ID}_{|k})$ by definition. We show that \mathcal{B} can compute all elements of this private key given the values that it knows. Recall that $h_j = g_{n+1-j}$. To generate a_0 , we first assume that $k < z$, and observe

$$\begin{aligned} a_0 &= g_i^u \left(\prod_{j \in S^*} g_{n+1-j+i} \right)^{-1} \cdot \underbrace{\left(g^\sigma \cdot \prod_{j=1}^{k-1} g_{n+1-j}^{I_j - I_j^*} \cdot g_{n+1-k}^{I_k - I_k^*} \cdot \prod_{j=k+1}^z g_{n+1-j}^{-I_j^*} \right)^{\tilde{s}}}_{=1} \\ &= g_i^u \left(\prod_{\substack{j \in S^* \\ j \neq i}} g_{n+1-j+i} \right)^{-1} \cdot \underbrace{g_{n+1}^{-1} \cdot g_{n+1-k}^{(I_k - I_k^*)\tilde{s}}}_{T_1} \cdot \underbrace{g^{\sigma\tilde{s}}}_{T_2} \cdot \underbrace{\prod_{j=k+1}^z g_{n+1-j}^{-I_j^* \tilde{s}}}_{T_3} \end{aligned}$$

The term T_1 can be computed by \mathcal{B} since

$$T_1 = g_{n+1}^{-1} \cdot g_{n+1-k}^{(I_k - I_k^*)(s + \frac{\alpha^k}{I_k - I_k^*})} = g_{n+1}^{-1} \cdot g_{n+1-k}^{(I_k - I_k^*)s} \cdot g_{n+1-k}^{\alpha^k} = g_{n+1-k}^{(I_k - I_k^*)s},$$

where the unknown term g_{n+1} is canceled out. The term T_2 can be computed by using g_k , which is not g_{n+1} since $k \leq z \leq L \leq n$. Each term in the product T_3 is computable since $g_{n+1-j}^{\tilde{s}} = g_{n+1-j}^s \cdot g_{n+1-j+k}^{1/(I_k - I_k^*)}$ and for $j = k+1, \dots, z$, the terms $g_{n+1-j}, g_{n+1-j+k}$ are not equal to g_{n+1} hence can be computed. It is left to consider the case $k = z$. In this case, a_0 is exactly the same as above except that the last product term, i.e., T_3 , does not appear. The analysis of computability by \mathcal{B} thus follows from the same argument.

The component a_1 can be generated since $a_1 = g^{\tilde{s}} = g^s \cdot g_k^{1/(I_k - I_k^*)}$. For $j = k+1, \dots, L$, the value b_j can be computed as $b_j = h_j^{\tilde{s}} = h_j^s \cdot g_{n+1-j+k}^{1/(I_k - I_k^*)}$.

Case 2: ID^* is a prefix of ID . Then it holds that $z+1 \leq w$. \mathcal{B} responds to the query by first computing a private key for node $(i, ID_{|z+1})$ from which it then construct a private key for the request node (i, ID) . \mathcal{B} picks random elements $s \in \mathbb{Z}_p$. We pose $\tilde{s} = s + \alpha^{z+1}/I_{z+1}$. Note that \tilde{s} is unknown to \mathcal{B} . Next, \mathcal{B} generates the private key in exactly the same form as Eq.(1) (change k to $z+1$, of course). From a similar observation as above, one can show that \mathcal{B} can compute this key.

Challenge. To generate the challenge, \mathcal{B} computes hdr^* as (h, h^u, h^σ) . It then randomly chooses a bit $b \in \{0, 1\}$ and sets $K_b = Z$ and picks a random K_{1-b} in \mathbb{G}_1 . \mathcal{B} then gives (hdr^*, K_0, K_1) to \mathcal{A} .

We claim that when $Z = e(g_{n+1}, h)$ (that is, the input to \mathcal{B} is a n -BDHE tuple) then (hdr^*, K_0, K_1) is a valid challenge to \mathcal{A} as in a real attack game. To see this, write $h = g^t$ for some (unknown) $t \in \mathbb{Z}_p$. Then, we have that

$$h^u = (g^u)^t = (g^u (\prod_{j \in S^*} g_{n+1-j})^{-1} (\prod_{j \in S^*} g_{n+1-j}))^t = (v \prod_{j \in S^*} g_{n+1-j})^t,$$

$$h^\sigma = \left(\prod_{j=1}^z g_{n+1-j}^{I_j^*} \cdot (g^\sigma \cdot \prod_{j=1}^z g_{n+1-j}^{-I_j^*}) \right)^t = (h_1^{I_1^*} \cdots h_z^{I_z^*} \cdot y)^t.$$

Thus, by definition, (h, h^u, h^σ) is a valid encryption of the key $e(g_{n+1}, g)^t$. Also, $e(g_{n+1}, g)^t = e(g_{n+1}, h) = Z = K_b$ and hence (hdr^*, K_0, K_1) is a valid challenge.

On the other hand, when Z is random in \mathbb{G}_1 (that is, the input to \mathcal{B} is a random tuple) then K_0, K_1 are just random independent elements of \mathbb{G}_1 .

Phase 2. \mathcal{A} continues to ask queries not issued in Phase 1. \mathcal{B} responds as before.

Guess. Finally, \mathcal{A} outputs $b' \in \{0, 1\}$. If $b = b'$ then \mathcal{B} outputs 1 (meaning $Z = e(g_{n+1}, h)$). Otherwise, it outputs 0 (meaning Z is random in \mathbb{G}_1).

We see that if $(g, h, \mathbf{y}_{g,\alpha,n}, Z)$ is sampled from \mathcal{R}_{BDHE} then $\Pr[\mathcal{B}(g, h, \mathbf{y}_{g,\alpha,n}, Z) = 0] = \frac{1}{2}$. On the other hand, if $(g, h, \mathbf{y}_{g,\alpha,n}, Z)$ is sampled from \mathcal{P}_{BDHE} then $|\Pr[\mathcal{B}(g, h, \mathbf{y}_{g,\alpha,n}, Z) = 0] - \frac{1}{2}| \geq \epsilon$. It follows that \mathcal{B} has advantage at least ϵ in solving n -BDHE problem in \mathbb{G} . This concludes the proof of Theorem 2. \square

4.3 Extensions

Modification. Recall that for BasicHICBE2 when $L > n$, we created dummy users so that the effective number of users is L . The resulting pk contained $2L+2$ elements of \mathbb{G} . We now give a more efficient scheme in this case ($L > n$). First, we change ‘ n ’ in all appearances in the description of BasicHICBE2 to ‘ L ’ except that the user indexes are as usual: $\{1, \dots, n\}$. Then we modify the public key to $\text{pk} = (g, g_1, \dots, g_L, g_{L+2}, \dots, g_{L+n}, v, y) \in \mathbb{G}^{L+n+2}$, which is of smaller size than that of the above method. This modified scheme is secure under the Decision L -BDHE assumption. However, it can be shown to be secure under a weaker one which is a new assumption that we call Decision $\langle L, n \rangle$ -BDHE. (Two values are specified instead of only one). It is defined exactly the same as the Decision L -BDHE except that we change $\mathbf{y}_{g,\alpha,L}$ to $\mathbf{y}_{g,\alpha,\langle L,n \rangle} := (g_1, \dots, g_L, g_{L+2}, \dots, g_{L+n})$.

Generalizations. Without going into details, we can also combine the BGW system with the Hybrid BB/BBG scheme [5, full §4.2], which can trade off the

public key and private key sizes with the ciphertext size. We denote this scheme by **BasicHICBE**(ω) for parameter $\omega \in [0, 1]$. It becomes **BasicHICBE1** when $\omega = 1$ and **BasicHICBE2** when $\omega = 0$. In this scheme, the public key, the private key, and the ciphertext contains $L^\omega + \max(L^{1-\omega}, n) + n + 1$, $\leq L^{1-\omega} + L^\omega + 1$, and $\leq L^\omega + 2$ elements in \mathbb{G} respectively. It can also be further generalized in the other dimension, namely the user dimension, in the same manner as the generalized BGW scheme [7], which can trade off the public key size with the ciphertext size while the private key size remains fixed. In the resulting scheme, denoted by **GenHICBE**(ω, μ), for $\mu \in [0, 1]$, the public key, the private key, and the ciphertext contains $L^\omega + \max(L^{1-\omega}, n^\mu) + n^\mu + n^{1-\mu}$, $\leq L^{1-\omega} + L^\omega + 1$, $\leq L^\omega + n^{1-\mu} + 1$ elements in \mathbb{G} respectively. Note that it becomes **BasicHICBE**(ω) when $\mu = 1$.

Chosen-Ciphertext and Adaptive-ID Security. We use the conversion due to Canetti et al. [13] or its derivatives [9,10] (adapted to the case of HICBE appropriately) to obtain **IND-sID-sSet-CCA**-secure schemes. An **IND-aID-sSet-CCA**-secure scheme can be constructed by combining the BGW system with Waters' HIBE [21] in essentially the same way as our previous two schemes.

5 Forward-Secure Public-Key Broadcast Encryption

Model for FS-BE. The syntax of a forward-secure public-key broadcast encryption (FS-BE) scheme is introduced in [22]. Following [7], for simplicity we define it as a KEM. A key-evolving broadcast encryption is made up of six randomized algorithms. Via $(\mathbf{pk}, \mathbf{msk}_0) \xleftarrow{R} \text{Setup}(n, T)$, where n is the number of receivers and T is the total number of time periods, the setup algorithm produces a public key \mathbf{pk} and an initial master private key \mathbf{msk}_0 ; via $\mathbf{msk}_{i,\tau} \xleftarrow{R} \text{MasUpdate}(\mathbf{pk}, \tau, \mathbf{msk}_{\tau-1})$ the master key update algorithm outputs a new private key $\mathbf{msk}_{i,\tau}$ of user i for time period τ ; via $\mathbf{sk}_{i,\tau} \xleftarrow{R} \text{Regist}(i, \tau, \mathbf{pk}, \mathbf{msk}_\tau)$ the center outputs a private key $\mathbf{sk}_{i,\tau}$ of user i for time period τ ; via $\mathbf{sk}_{i,\tau} \xleftarrow{R} \text{Update}(\mathbf{pk}, i, \tau, \mathbf{sk}_{i,\tau-1})$ the user i updates his private key to $\mathbf{sk}_{i,\tau}$ for the consecutive time period; via $(\text{hdr}, K) \xleftarrow{R} \text{Encrypt}(\mathbf{pk}, S, \tau)$, where S is the set of recipients, a sender outputs a pair (hdr, K) , a header and a message encryption key; via $K \xleftarrow{R} \text{Decrypt}(\mathbf{pk}, S, i, \mathbf{sk}_{i,\tau}, \text{hdr})$ a recipient $i \in S$ outputs $K \in \mathcal{K}$. A scheme is correct if (1) when $\mathbf{pk}, \mathbf{msk}_\tau, \mathbf{sk}_{i,\tau-1}$ are correctly generated, the distributions of private keys output from $\text{Regist}(i, \tau, \mathbf{pk}, \mathbf{msk}_\tau)$ and from $\text{Update}(\mathbf{pk}, i, \tau, \mathbf{sk}_{i,\tau-1})$ are the same; (2) **Encrypt** and **Decrypt** are consistent (in the standard way).

Security Notions. We define semantic security of a key-evolving BE in essentially the same way as in the case of HICBE system. Such a notion is introduced by Yao et al. [22]. We reformatize and briefly state it here. (See the full paper [2] for details). We define eight combinations of notions called **IND-xFS_i-ySet-CCA** security where $(x, y) \in \{(a, a), (a, s), (s, a), (s, s)\}$, corresponding to whether the target time τ^* and/or the target set of recipients S^* must be disclosed before the Setup phase or not, and $i \in \{1, 2\}$, where when $i = 2$ the adversary is allowed to ask also master key queries for \mathbf{msk}_τ of time τ while when $i = 1$ it

is not. Note that the notion in [22] corresponds to IND-aFS₁-aSet-CCA security. We note that a IND-sFS_{*i*}-ySet-CCA-secure scheme is also secure in the sense IND-aFS_{*i*}-ySet-CCA, albeit with the security degradation by factor T . For most applications, FS₁ security is sufficient. In this case, it is useful to consider the MasUpdate as a trivial algorithm as we let $\text{msk}_\tau = \text{msk}_0$ for all τ (and denote it by msk). Note that it is trivial to convert a scheme with FS₁ security to a new one achieving FS₂ security by letting msk_τ contains all user keys of time τ .

Conversion C [HICBE⇒FS-BE]. Given a HICBE scheme, we construct a FS-BE scheme using the “time tree” technique of [12], which was used to construct a forward-secure encryption from a binary tree encryption. Our conversion is essentially the same as that of [12] except that the user dimension is introduced.

For a forward-secure BE with T time periods, we image a complete balance binary tree of depth $L = \log_2(T + 1) - 1$. Let each node be labeled with a string in $\{0, 1\}^{\leq L}$. We assign the root with the empty string. The left and right child of w is labeled $w0$ and $w1$ respectively. From now, to distinguish the abstract ‘node’ of a HICBE system from nodes in the binary tree, we refer to the former as h-node and the latter as usual. Following the notation in [12], we let w^τ to be the τ -th node in a pre-order traversal of the binary tree.⁵ WLOG, we assume that $0, 1 \in \mathcal{I}$, the identity space. Hence, we can view a binary string of length $z \leq L$ as an identity tuple of length z . Encryption in time τ for a set S of recipients uses the encryption function of the HICBE scheme to the multi-node (S, w^τ) . At time τ the private key also contains, beside the private key of h-node (i, w^τ) of the HICBE scheme, all the keys of h-nodes (i, y) where y is a right sibling of the nodes on the path from the root to w^τ in the binary tree. When updating the key to time $\tau + 1$, we compute the private key of h-node $(i, w^{\tau+1})$ and erase the one of (i, w^τ) . Since $w^{\tau+1}$ is a left child of w^τ or one of the nodes whose keys are stored as the additional keys at time τ , the derivation can be done, in particular, using at most one application of Derive. We denote this conversion as $C(\cdot)$ and write its formal description and its security proof in [2].

Theorem 3. *Suppose that the scheme HICBE for L levels is (t, q_P, q_D, ϵ) -IND-xID-ySet-CCA-secure (resp., (t, q_P, ϵ) -IND-xID-ySet-CPA-secure) for some $(x, y) \in \{(a, a), (a, s), (s, a), (s, s)\}$. Then the scheme $C(\text{HICBE})$ for T time periods is (t, q'_P, q_D, ϵ) -IND-xFS₁-ySet-CCA-secure (resp., (t, q'_P, ϵ) -IND-xFS₁-ySet-CPA-secure) for $q'_P \leq q_P/L$, where $L = \log(T + 1) - 1$.*

Resulting FS-BE Schemes. It is easy to see that in the resulting scheme, the private key size is expanded by the factor $O(\log T)$ while the other parameters are unchanged from the original HICBE scheme (instantiated for $\log(T + 1) - 1$ levels of identities). We have that the $C(\text{BasicHICBE1})$ scheme achieves ciphertext of size $O(\log T)$ and user private keys of size $O(\log^2 T)$ while the $C(\text{BasicHICBE2})$ scheme achieves ciphertexts of size $O(1)$ and user private keys of size $O(\log^2 T)$.

We also directly construct a more efficient but specific FS-BE scheme, denoted by DirFSBE, which is not built via the generic conversion. It can be considered as

⁵ The pre-order traversal is started from the root, $w^1 = \varepsilon$. From w it goes to $w0$ if w is not a leaf otherwise it goes to $v1$ if $v0$ is the largest string that is a prefix of w .

Table 1. Comparison among previous and our FS-BE schemes (upper and lower table resp.). $T = |\text{total time periods}|$. $n = |\text{all users}|$. $r = |\text{revoked users}|$. The time complexity is expressed in terms of number of operations where [e] is exponentiation, [p] is bilinear pairing, and [m] is group multiplication, while [o] indicates the time complexity for some other process. ‘ \Leftarrow ’ means that it has the same value as the entry on its left.

Params↓	$\text{GS}_{(\text{NNL})} \times_{\text{YFDL}} \text{GS}$ [22]	$\text{BBG}_{(\text{NNL})} \times_{\text{YFDL}} \text{BBG}$ [5, full §5.2]	$\text{BBG}_{(\text{NNL})} \perp_{\text{BBG}} \text{BB}$ [5, full §C]
Reg time	$O(\log^3 n \log T)$ [e]	\Leftarrow	$O((\log^2 n)(\log n + \log T))$ [e]
Enc time	$O(r \log n \log T)$ [e]	\Leftarrow	$O(r(\log n + \log T))$ [e]
Dec time	$O(\log n \log T)$ [p] + $O(r)$ [o]	\Leftarrow	$O(\log T)$ [p] + $O(r)$ [o]
Upd time	$O(\log^3 n)$ [e]	\Leftarrow	$O(\log^2 n \log T)$ [e]
Pub key	$O(\log n + \log T)$	\Leftarrow	\Leftarrow
Pri key	$O(\log^3 n \log T)$	\Leftarrow	$O((\log^2 n)(\log n + \log T))$
Cipher	$O(r \log n \log T)$	$O(r)$	$O(r \log T)$

Params↓	C(BasicHICBE1)	DirFSBE	C(BasicHICBE2)	C(GenHICBE(0.5, 0.5))
Reg time	$O(\log T)$ [e]	\Leftarrow	\Leftarrow	$O(\sqrt{\log T})$ [e]
Enc time	$O(n)$ [m] + $O(\log T)$ [e]	\Leftarrow	\Leftarrow	$O(\sqrt{n})$ [m] + $O(\sqrt{\log T})$ [e]
Dec time	$O(n)$ [m] ⁶ + $O(\log T)$ [p]	\Leftarrow	$O(n)$ [m] ⁶ + $O(1)$ [p]	$O(\sqrt{n})$ [m] + $O(\sqrt{\log T})$ [p]
Upd time	$O(1)$ [e]	\Leftarrow	\Leftarrow	\Leftarrow
Pub key	$O(n + \log T)$	\Leftarrow	\Leftarrow	$O(\sqrt{n} + \sqrt{\log T})$
Pri key	$O(\log^2 T)$	$O(\log T)$	$O(\log^2 T)$	$O(\log^{1.5} T)$
Cipher	$O(\log T)$	\Leftarrow	$O(1)$	$O(\sqrt{n} + \sqrt{\log T})$

a redundancy-free version of C(BasicHICBE1) which can reduce private key size to $O(\log T)$ without affecting other parameters. This can be seen as a reminiscent of the “Linear fs-HIBE” scheme in [5, full §C]. Its generalized scheme, denoted by DirFSBE(μ), can be constructed as in §4.3. It trades off the public keys of size $O(n^\mu + n^{1-\mu} + \log T)$ with the ciphertexts of size $O(n^{1-\mu} + \log T)$.

Efficiency Comparisons. We draw comparisons among FS-BE schemes by wrapping up in Table 1. We name the three previous schemes intuitively from their approaches, where ‘ \times_{YFDL} ’ is the “cross-product” approach by Yao et al. [22], ‘ \perp_{BBG} ’ is the orthogonal integration approach by Boneh et al. [5, full §C], and the two operands indicate the underlying HIBEs, which include GS (the Gentry-Silverberg HIBE [17]), BB, and BBG. (See more details in [2]).

6 Public-Key Broadcast Encryption with Keyword Search

6.1 Definitions and Relation to Anonymous ICBE

Model for BEKS. A public-key BE with keyword search (BEKS) consists of four algorithms. Via $(\text{pk}, \{\text{sk}_1, \dots, \text{sk}_n\}) \xleftarrow{R} \text{Setup}(n)$ the setup algorithm

⁶ This is due to the computation of $\prod_{j \in S, j \neq i} g_{n+1-j+i}$, which indeed can be pre-computed. This is useful when S is incrementally changed (cf. [7]).

produces a public key and n user keys; via $C \xleftarrow{R} \text{BEKS}(\text{pk}, S, w)$ a sender encrypts a keyword w to get a ciphertext (C, S) intended for recipients in $S \subseteq \{1, \dots, n\}$; via $t_{i,w} \xleftarrow{R} \text{Td}(i, w, \text{sk}_i)$ the receiver i computes a trapdoor $(t_{i,w}, i)$ for keyword w and provides it to the gateway (the server); via $b \leftarrow \text{Test}(\text{pk}, i, t_{i,w}, C, S)$ for $i \in S$ the gateway can test whether C encrypts w where $b = 1$ means “positive” and $b = 0$ means “negative”. Here if $i \notin S$ it always outputs ‘ \notin ’. We describe the right-keyword consistency (correctness), the computational consistency (in the sense of [1]), and the security notion, which we name IND-xKW-ySet-CPA, in the full paper [2]. The security captures the property that the adversary be unable to distinguish the encryption of chosen keyword with a random one.

Conversion K [ICBE \Rightarrow BEKS]. The conversion of [1] that compiles any anonymous IBE into a PEKS can be generalized to a broadcast version straightforwardly. More concretely, we construct BEKS from ICBE as follows. $\text{Setup}_{\text{BEKS}}(n)$ can be constructed from $\text{Setup}_{\text{ICBE}}$ and $\text{PrivKeyGen}_{\text{ICBE}}$ by relating the same public key pk and relating the private key $\text{sk}_i = d_i$. The remaining algorithms work as follows: $t_{i,w} \xleftarrow{R} \text{Td}(i, w, \text{sk}_i) = \text{Derive}_{\text{ICBE}}(i, w, d_i)$; $(C_1, C_2) \xleftarrow{R} \text{BEKS}(\text{pk}, S, w) = \text{Encrypt}_{\text{ICBE}}(\text{pk}, S, w)$; $\text{Test}(\text{pk}, i, t_w, (C_1, C_2), S)$ outputs ‘ \notin ’ if $i \notin S$, else outputs 1 if $\text{Decrypt}_{\text{ICBE}}(\text{pk}, S, i, t_w, C_1) = C_2$, else outputs 0. Denote this conversion by $K(\cdot)$. Its correctness is immediate from that of ICBE. Indeed, $t_{i,w}, C_1, C_2$ are related to $d_{i,w}, \text{hdr}, K$ in the ICBE scheme respectively. We remark that our conversion is a little bit different from (and simpler than) that of [1], in particular, since we have formalized the ICBE as a KEM.

Theorem 4. (Informal) *If the scheme ICBE is ANO-xID-ySet-CPA[$\{1\}$]-secure, then the BEKS scheme $K(\text{ICBE})$ is IND-xKW-ySet-CPA-secure. Further, if ICBE is semantically secure, then $K(\text{ICBE})$ is computationally consistent.*

6.2 Constructing Anonymous (H)ICBE

Attempts. As one may expect, the first attempt is to use our integration method to combine the BGW system with the anonymous HIBE, BW, by Boyen-Waters [11], which has a BB/BBG-like structure. Somewhat surprisingly and unfortunately, the resulting HICBE scheme is *not* ANO-sID-sSet-CPA-secure. Essentially, this is precisely due to the implicit orthogonality of BGW and BW. Such a property enables any user $i \notin S^*$ to use the independent part of private keys corresponding to the BW portion to easily distinguish whether a ciphertext is intended for (S^*, ID^*) or (S^*, R) for random R , thus breaking anonymity. Dilemmatically, on the one hand, this orthogonality enables us to prove the confidentiality of the combined scheme; on the other hand, this very property gives an attack to the anonymity. We also remark that the approach $\text{BB}_{(\text{NNL})} \perp_{\text{BBG}} \text{BW}$ and $\text{BBG}_{(\text{NNL})} \perp_{\text{BBG}} \text{BW}$ (where notations are borrowed from the end of §5) also do *not* preserve the anonymity of BW due to a similar reason. See details in [2].

The Construction. From the above discussion, it is then natural to implement both the broadcast and identity dimensions from two non-orthogonal sub-

systems. Therefore, we construct our scheme in [2], denoted by AnonHICBE, from the YFDL (cross-product) approach instantiated to two copies of the BW hierarchies, or in our terminology, $BW_{(NNL)} \times_{YFDL} BW$.⁷ The resulting anonymous ICBE system achieves ciphertext of size $O(r \log n)$ and private key of size $O(\log^4 n)$ for the user level (level 0) and private key of size $O(\log^3 n)$ for level 1. These translate to the sizes of ciphertext, private key, and trapdoor in BEKS respectively.

References

1. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Advances in Cryptology — CRYPTO 2005*, LNCS 3621, pp. 205-222. Springer, 2005.
2. N. Attrapadung, J. Furukawa, and H. Imai. Full version of this paper (with the same title). To be available at <http://eprint.iacr.org>.
3. N. Attrapadung and H. Imai. Graph-decomposition-based frameworks for subset-cover broadcast encryption and efficient instantiations. In *Advances in Cryptology — Asiacrypt 2005*, LNCS 3788 of LNCS, pp. 100-120. Springer, 2005.
4. D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology — Eurocrypt 2004*, LNCS 3027 of LNCS, pp. 223-238. Springer, 2004.
5. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology — Eurocrypt 2005*, LNCS 3494, pp. 440-456. Springer, 2005. Full version available at <http://eprint.iacr.org/2005/015>.
6. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology — Eurocrypt 2004*, LNCS 3027, pp. 506-522. Springer, 2004.
7. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology — Crypto 2005*, LNCS 3621, pp. 258-275. Springer, 2005.
8. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto 2001*, LNCS 2139, pp. 213-229. Springer, 2001.
9. D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In *RSA-CT 2005*, LNCS 3376, pp. 87-103. 2005.
10. X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In *Proc. ACM-CCS 2005*, pp. 320-329, ACM Press, 2005.
11. X. Boyen, B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology — Crypto 2006*, LNCS 4117, pp. 290-307. Springer, 2006. Full version available at <http://eprint.iacr.org/2006/085>.
12. R. Canetti, S. Halevi, J. Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology — Eurocrypt 2003*, LNCS 2656, pp. 255-271. 2003.

⁷ Probably this is the same approach as the one that Boyen and Waters used to construct an anonymous FS-HIBE, briefly mentioned in [11, full p.4], although it might be a different one (and we would not know since no detail was given there). One difference to mention is that the two hierarchies correspond to time/ID there, as opposed to broadcast/ID here.

13. R. Canetti, S. Halevi, J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology — Eurocrypt 2004*, LNCS 3027, pp. 207-222.
14. J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology — Eurocrypt 2006*, LNCS 4004, pp. 1-11, Springer, 2006.
15. Y. Dodis and N. Fazio. Public-key broadcast encryption for stateless receivers. In *ACM Digital Rights Management 2002*, LNCS 2696, pp. 61-80. Springer, 2002.
16. A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology — Crypto 1993*, LNCS 773, pp. 480-491. Springer, 1993.
17. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology — Asiacrypt 2002*, LNCS 2501, pp. 548-566. Springer, 2002.
18. M. T. Goodrich, J. Z. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Advances in Cryptology — Crypto 2004*, LNCS 3152, pp. 511-527. Springer, 2004.
19. D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In *Advances in Cryptology — Crypto 2002*, LNCS 2442, pp. 47-60. Springer, 2002.
20. D. Naor, M. Naor, J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology — Crypto 2001*, LNCS 2139, pp. 41-62. 2001.
21. B. Waters. Efficient identity-Based encryption without random oracles. In *Advances in Cryptology — Eurocrypt 2005*, LNCS 3494, pp. 114-127. Springer, 2005.
22. D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proc. ACM-CCS 2004*, pp. 354-363, ACM, 2004.

On the Generic Construction of Identity-Based Signatures with Additional Properties

David Galindo¹, Javier Herranz², and Eike Kiltz²

¹ Institute for Computing and Information Sciences,
Radboud University, Nijmegen, The Netherlands

² Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands
d.galindo@cs.ru.nl, j.herranz@cwi.nl, kiltz@cwi.nl

Abstract. It has been demonstrated by Bellare, Neven, and Namprempre (Eurocrypt 2004) that identity-based signature schemes can be constructed from any PKI-based signature scheme. In this paper we consider the following natural extension: is there a generic construction of “identity-based signature schemes with additional properties” (such as identity-based blind signatures, verifiably encrypted signatures, ...) from PKI-based signature schemes with the same properties? Our results show that this is possible for great number of properties including proxy signatures; (partially) blind signatures; verifiably encrypted signatures; undeniable signatures; forward-secure signatures; (strongly) key insulated signatures; online/offline signatures; threshold signatures; and (with some limitations) aggregate signatures.

Using well-known results for PKI-based schemes, we conclude that such identity-based signature schemes with additional properties can be constructed, enjoying some better properties than specific schemes proposed until now. In particular, our work implies the existence of identity-based signatures with additional properties that are provably secure in the standard model, do not need bilinear pairings, or can be based on general assumptions.

1 Introduction

Digital signatures are one of the most fundamental concepts of modern cryptography. They provide authentication, integrity and non-repudiation to digital communications, which makes them the most used public key cryptographic tool in real applications. In order to satisfy the needs of some specific scenarios such as electronic commerce, cash, voting, or auctions, the original concept of digital signature has been extended and modified in multiple ways, giving rise to many kinds of what we call “digital signatures with additional properties”, e.g. blind signatures, verifiably encrypted signatures, and aggregated signatures.

Initially, all these extensions were introduced for the standard PKI-based framework, where each user generates a secret key and publishes the matching public key. In practice, digital certificates linking public keys with identities of users are needed to implement these systems, and this fact leads to some drawbacks in efficiency and simplicity. For this reason, the alternative framework of

identity-based cryptography was introduced by Shamir [29]. The idea is that the public key of a user can be directly derived from his identity, and therefore digital certificates are avoidable. The user obtains his secret key by interacting with some trusted master entity. In his paper, Shamir already proposed an identity-based signature scheme. In contrast, the problem of designing an efficient and secure identity-based encryption scheme remained open until [6,28].

From a theoretical point of view, results concerning identity-based encryption schemes are more challenging than those concerning identity-based signatures (IBS). In contrast to the identity-based encryption case it is folklore that a standard PKI-based signature scheme already implies an identity-based signature scheme by using the signature scheme twice: for generating user secret keys and for the actual signing process. More precisely, the user secret key of an identity consists of a fresh PKI-based signing/verification key and a certificate proving the validity of the signing key. The latter certificate is established by the master entity by signing (using the master signing key) the new verification key together with the user's identity. In the actual identity-based signing process the user employs this signing key to sign the message. The identity-based signature itself consists of this signature along with the certificate and the public verification key.

The above idea was formalized by Bellare, Neven, and Namprempre in [3], where they propose a generic and secure construction of identity-based signature schemes from any secure PKI-based signature scheme. However, some specific identity-based signature schemes have been proposed and published, mostly employing bilinear pairings and random oracles, without arguing if the proposed schemes are more efficient than the schemes resulting from the generic construction in [3]. In fact, in many papers the authors do not mention the generic approach from [3] and in spite of Shamir's work from more than two decades ago [29] it still seems to be a popular "opinion" among some researchers that the construction of identity-based signatures inherently relies on bilinear pairings.

Our observation is that the situation is quite similar when identity-based signature schemes with additional properties are considered. Intuitively such schemes may be obtained using the same generic approach as in the case of standard identity-based signatures combining a digital certificate and a PKI-based signature scheme with the desired additional property. To the best of our knowledge, this intuitive construction was never mentioned before, nor has a formal analysis been given up to now. Furthermore, specific identity-based signature schemes with additional properties keep being proposed and published without arguing which improvements they bring with respect to the possible generic certificate-based approach. Nearly all of these papers employ bilinear pairings and the security proofs are given in the random oracle model [5] (with its well-known limitations [9]).

1.1 Our Results

In this work we formally revisit this intuitive idea outlined in the last paragraph. Namely, if \mathcal{S} is a secure PKI-based signature scheme and \mathcal{PS} is a PKI-based signature scheme with some additional property \mathcal{P} , we pursue the question if for a

certain property \mathcal{P} the combination of those two signature schemes can lead to a secure IBS scheme IB_PS enjoying the same additional property \mathcal{P} . We can answer this question to the positive, giving generic constructions of signature schemes with the following properties: proxy signatures (PS); (partially) blind signatures (BS); verifiably encrypted signatures (VES); undeniable signatures (US); forward-secure signatures (FSS); strong key insulated signatures (SKIS); online/offline signatures (OOS); threshold signatures (TS); and aggregate signatures (AS).¹

IMPLICATIONS. By considering well-known results and constructions of PKI-based signatures PS with the required additional properties, we obtain identity-based schemes IB_PS from weaker assumptions than previously known. A detailed overview of our results can be looked up in Table 1 on page 183. To give a quick overview of our results, for nearly every property \mathcal{P} listed above, we obtain (i) the first IB_PS scheme secure in the standard model (i.e., without random oracles); (ii) the first IB_PS scheme built without using bilinear pairings; and (iii) the first IB_PS based on “general assumptions” (e.g. on the sole assumption of one-way functions), answering the main foundational question with regard to these primitives. Our results therefore implicitly resolve many “open problems” in the area of identity-based signatures with additional properties.

GENERIC CONSTRUCTIONS. For some properties \mathcal{P} the construction of the scheme IB_PS is the same as in [3] and a formal security statement can be proved following basically verbatim the proofs given in [3]. But as the limitations of the generic approach indicate, this approach does not work in a black-box way for every possible property \mathcal{P} . For some special properties the certificate-based generic construction sketched above has to be (non-trivially) adapted to fit the specific nature of the signature scheme. This is in particular the case for blind and undeniable signatures and hence in these cases we will lay out our constructions in more detail.

DISCUSSION. We think that in some cases the constructions of identity-based signatures with additional properties implied by our results are at least as efficient as most of the schemes known before. However, because of the huge number of cases to be considered, we decided not to include a detailed efficiency analysis of our generic constructions. Note that, in order to analyze the efficiency of a particular identity-based scheme resulting from our construction, we should first fix the framework: whether we admit the random oracle model, whether we allow the use of bilinear pairings, etc. Then we should take the most efficient suitable PKI-based scheme and measure the efficiency of the resulting identity-based one. Our point is rather that this comparison should be up to the authors proposing new specific schemes: the schemes (explicitly and implicitly) implied by our generic approach should be used as benchmarks relative to which both, existing and new practical schemes measure their novelty and efficiency.

¹ We stress that the length of our implied aggregated identity-based signatures is still depending linearly on the number of different signers (optimally it is constant) and therefore our results concerning AS are not optimal.

We stress that we do not claim the completely novelty of our generic approaches to construct identity-based signatures with additional properties. Similar to [3] we rather think that most of these constructions can be considered as folklore and are known by many researchers. However, the immense number of existing articles neglecting these constructions was our initial motivation for writing this paper. We think that our results may also help better understanding IBS. To obtain a practical IBS with some additional properties the “standard method” in most articles is to start from a standard IBS and try to “add in” the desired additional property. Our results propose that one should rather start from a standard signature scheme with the additional property and try to make it identity-based. We hope that the latter approach may be used to obtain more efficient practical schemes.

2 Definitions

STANDARD SIGNATURES. A standard signature scheme $\mathcal{S} = (\text{S.KG}, \text{S.Sign}, \text{S.Vfy})$ consists of the following three (probabilistic polynomial-time) algorithms. The **key generation** algorithm S.KG takes as input a security parameter k and returns a secret key SK and a matching public key PK . We use the notation $(SK, PK) \leftarrow \text{S.KG}(1^k)$ to refer to one execution of this protocol. The **signing** algorithm S.Sign inputs a message m and a secret key SK . The output is a signature $sig_{SK}(m)$. We denote an execution of this protocol as $sig_{SK}(m) \leftarrow \text{S.Sign}(SK, m)$. The **verification** algorithm S.Vfy takes as input a message m , a signature $sig = sig_{SK}(m)$ and a public key PK . The output is 1 if the signature is valid, or 0 otherwise. We use the notation $\{0, 1\} \leftarrow \text{S.Vfy}(PK, m, sig)$ to refer to one execution of this algorithm.

The standard security notion for signature schemes in unforgeability against adaptively-chosen message attacks, which can be found in [19,17].

IDENTITY-BASED SIGNATURES. An identity-based signature scheme $IB_S = (\text{IB_S.KG}, \text{IB_S.Extr}, \text{IB_S.Sign}, \text{IB_S.Vfy})$ consists of the following four (probabilistic polynomial-time) algorithms [10]. The **setup** algorithm IB_S.KG takes as input a security parameter k and returns, on the one hand, the system public parameters mpk and, on the other hand, the value master secret key msk , which is known only to the master entity. We note an execution of this protocol as $(mpk, msk) \leftarrow \text{IB_S.KG}(1^k)$. The **key extraction** algorithm IB_S.Extr takes as inputs mpk , the master secret key msk and an identity $id \in \{0, 1\}^*$, and returns a secret key $sk[id]$ for the user with this identity. We use notation $sk[id] \leftarrow \text{IB_S.Extr}(msk, id)$ to refer to one execution of this protocol. The **signing** algorithm IB_S.Sign inputs a user secret key $sk[id]$, the public parameters mpk , an identity, and a message m . The output is a signature $sig = sig_{msk}(id, m)$. We denote an execution of this protocol as $sig \leftarrow \text{IB_S.Sign}(mpk, id, sk[id], m)$. Finally, the **verification** algorithm IB_S.Vfy inputs mpk , a message m , an identity id and a signature sig ; it outputs 1 if the signature is valid, and 0 otherwise. To refer to one execution of this protocol, we use notation $\{0, 1\} \leftarrow \text{IB_S.Vfy}(mpk, id, m, sig)$.

The standard security notion for identity-based signature schemes is unforgeability against adaptively-chosen identity and message attacks, which can be found in [3,17].

3 Generic Construction of Identity-Based Signatures

We first outline the BNN generic transformation [3] from two standard signature schemes \mathcal{S} , \mathcal{S}' into an identity-based signature scheme.

Let $\mathcal{S} = (\text{S.KG}, \text{S.Sign}, \text{S.Vfy})$ and $\mathcal{S}' = (\text{S'.KG}, \text{S'.Sign}, \text{S'.Vfy})$ be two (possibly equal) standard signature schemes. The generic construction of an identity-based signature scheme $\text{IB}_{\mathcal{S}} = (\text{IB.S.KG}, \text{IB.S.Extr}, \text{IB.S.Sign}, \text{IB.S.Vfy})$, proposed in [3], is defined as follows.

KEY GENERATION $\text{IB.S.KG}(1^k)$: The key generation algorithm from the standard signature scheme \mathcal{S} is run to obtain the master key-pair for the identity-based signature scheme $\text{IB}_{\mathcal{S}}: (msk, mpk) \leftarrow \text{S.KG}(1^k)$.

IBS KEY EXTRACTION $\text{IB.S.Extr}(msk, id_i)$: The secret key of a user with identity id_i is defined as

$$sk[id_i] = (sig_i, pk_i, sk_i), \quad (1)$$

where (pk_i, sk_i) is a random key-pair obtained by running $\text{S'.KG}(1^k)$ and $sig_i \leftarrow \text{S.Sign}(msk, id_i || pk_i)$. Here the signature sig_i can be viewed as a “certificate” on the validity of pk_i .

IDENTITY-BASED SIGN $\text{IB.S.Sign}(mpk, id_i, sk[id_i], m)$: Given a user secret key for id_i an id-based signature for identity id_i and message m is defined as

$$sig(id_i, m) = (sig_i, pk_i, sig_{sk_i}(m)), \quad (2)$$

where $sig_{sk_i}(m) = \text{S'.Sign}(sk_i, m)$ can be computed by the possessor of the user secret key $sk[id_i]$ since sk_i is contained in $sk[id_i]$. Signature sig_i included in Eqn. (2) certifies the validity of pk_i .

VERIFICATION $\text{IB.S.Vfy}(mpk, sig)$: The user checks if the first signature from Eqn. (2) is valid with respect to mpk and “message” $id || pk_i$ (using the verification protocol S.Vfy); and if the second signature is valid with respect to pk_i and the message m (using the verification protocol S'.Vfy).

Bellare, Namprempre, and Neven [3] prove the following result:

Theorem 1. *If \mathcal{S} and \mathcal{S}' are both secure standard signature schemes then $\text{IB}_{\mathcal{S}}$ is a secure identity-based signature scheme.*

Let \mathcal{PS} be a signature scheme with the property \mathcal{P} . We extend the above construction to an IBS with additional properties $\text{IB}_{\mathcal{PS}}$ in a straightforward way: as with signing/verification, all functionality provided by \mathcal{PS} is “lifted” to the identity-based case. That means that (analog to IB.S.Sign and IB.S.Vfy) any protocol additionally provided by \mathcal{PS} is executed using the corresponding secret/public key pair (sk_i, pk_i) from the user secret key Eqn. (1). We will refer to the latter construction as the “generic construction of identity-based signatures with additional properties” or simply “generic construction”.

In the rest of this section we will demonstrate that this generic construction and variants of it can indeed be used for many signatures schemes with additional properties. Due to the lack of space we only provide details for identity-based VES, US, AS, and BS schemes. For the details on the remaining results we refer to the full verion of this paper [17]. Table 1 summarizes the practical impact of our results, i.e. it is shown which types *IB-PS* of new identity-based signature schemes are implied by our general constructions. The existence of the identity-based signature schemes can be derived by the existence of the respective standard signature scheme [17].

Table 1. A summary of the practical implications of our results. Here “★” means that a scheme was known before (with a formal proof), a “★” means that our construction gives the first such scheme, and a “–” means that no such scheme is known.

Signature type	Existence of identity-based signature schemes			
	at all ?	w/o random oracles?	w/o pairings?	general assumptions?
VES §3.1	★	★	★	★
BS §4	★/★ ²	★	★	★
US §3.2	★	★	★	–
FSS [17]	★	★	★	★
SKIS [17]	★	★	★	★
PS [17]	★	★	★	★
OOS [17]	★	★	★	★
Threshold [17]	★	★	★	–

3.1 Verifiably Encrypted Signatures

Verifiably encrypted signature (VES) schemes can be seen as a special extension of the standard signature primitive. VES schemes enable a user Alice to create a signature encrypted using an adjudicator’s public key (the VES signature), and enable public verification if the encrypted signature is valid. The adjudicator is a trusted third party, who can reveal the standard signature when needed. VES schemes provide an efficient way to enable fairness in many practical applications such as contract signing.

An efficient VES scheme in the random oracle model based on pairings was given in [7], one in the standard model in [25]. It was further noted in [25] that VES schemes can be constructed on general assumptions such as trapdoor one-way permutations.

Identity-based verifiably encrypted signature (IB-VES) schemes were introduced in [20] where also a concrete security model was proposed. In contrast to [20], here we only consider a weaker (but still reasonable) model where the adjudicator has a fixed public key, i.e. it is not identity-based.

Compared to a standard signature a VES scheme has three additional algorithms: VES signing/verification (with respect to an adjudicators public key), and adjudication. Here the adjudication algorithm inputs an adjudicators secret

² Against concurrent adversaries.

key and transforms a VES into a standard signature. For our generic construction VES signing and verification can be lifted to the identity-based case in the same way as in the generic construction, i.e. in an IB-VES one replaces $\text{sig}_{sk_i}(m)$ in Eqn. (2) with its VES counterpart obtained by running the VES signing algorithm on sk_i , m , and the adjudicator’s public key. IB-VES verification checks the certificate and the VES using the standard VES verification algorithm. More formally we can prove the following theorem:

Theorem 2. *If S is a secure standard signature scheme and \mathcal{PS} is a secure verifiably encrypted signature scheme then the generic construction gives a secure identity-based verifiably encrypted signature scheme.*

Using our generic construction we get an IB-VES scheme based on any trapdoor one-way function [25], and a more efficient one using [7].

3.2 Undeniable Signatures

Undeniable signatures [12] (US) are signature schemes in which testing for (in)validity of a signature requires interaction with the signer. Undeniable signatures are used in applications where signed documents carry some private information about the signer and where it is considered to be an important privacy factor to limit the ability of verification.

Following [14], an undeniable signature scheme \mathcal{US} consists of four algorithms $\mathcal{US} = (\text{US.KG}, \text{US.Sign}, \text{US.Conf}, \text{US.Disav})$, where US.Conf is a confirmation and US.Disav is a disavowal protocol, both being interactive algorithms run between a prover and a verifier. The basic security properties are (standard) *unforgeability*, *non-transferability* and *simulatability*. By non-transferability it is meant that no adversary should be able to convince any third party of the validity/invalidity of a given message/signature pair after having participated in the confirmation and disavowal protocols. Intuitively this is captured by requiring the confirmation and disavowal protocols to be “zero-knowledge”, such that no information is leaked besides (in)validity. With simulatability one wants to ensure that the strings representing signatures can not be recognized (i.e., distinguished from a random string) by an attacker. This security property is fulfilled if there exists a signature simulator algorithm US.Sim , that on input of a public key and a message, outputs a simulated signature $\text{sig}(m)$ which looks like a “real undeniable signature” to anyone who only knows public information and has access to confirmation/disavowal oracles.

Extending the previous definition to the identity-based setting, an identity-based undeniable signature (IB-US) scheme consists of a tuple of five algorithms $\text{IB_US} = (\text{IB_US.KG}, \text{IB_US.Extr}, \text{IB_US.Sign}, \text{IB_US.Conf}, \text{IB_US.Disav})$ where IB_US.Conf and IB_US.Disav are interactive algorithms run between a prover and a verifier. The basic security properties for an IB-US (unforgeability, non-transferability and simulatability), are defined by suitably adapting the standard US security notions to the identity-based scenario.

In particular, the *identity-based simulatability* property is defined in terms of the existence of an additional simulation algorithm IB_US.Sim . On input of

the system public parameters mpk , an identity id and a message m , IB_US.Sim outputs a simulated signature $\text{sig}(id, m)$, which is indistinguishable from a real signature for someone having access to confirmation/disavowal oracles for the identity id .

We now sketch our generic construction of identity-based undeniable signatures. In contrast to the generic construction (cf. Eqn. (2)) we define the identity-based undeniable signature $\text{IB_US.Sign}(sk[id_i], m)$ as $\text{sig}_{sk_i}(m)$ (i.e., the certificate $\text{sig}_{msk}(id_i||pk_i)$ and pk_i are not included in the signature). In the interactive identity-based confirmation and disavowal protocols, the signer sends his certificate $(\text{sig}_{msk}(id_i||pk_i), pk_i)$ to the verifier such that the verifier can be convinced about the link between the signature and $id_i||pk_i$. Then prover (using sk_i) and verifier (using pk_i) engage in the standard US confirmation/disavowal protocol.

It remains to describe the identity-based simulation algorithm IB_US.Sim in terms of the algorithm US.Sim . We define the output of $\text{IB_US.Sim}(mpk, id, m)$ as $\text{US.Sim}(pk'_i, m)$, where $(pk'_i, sk'_i) \leftarrow \text{US.KG}(1^k)$ is a fresh key pair generated by the simulator. Note that the simulator IB_US.Sim does not input the user secret key $sk[id]$ and therefore the public key pk_i from the user secret key for id_i (cf. Eqn. (1)) is information theoretically hidden from it. However, an adversary against simulatability may learn this public key pk_i from an execution of the confirmation/disavowal protocol. It turns out that to ensure that our generic IB-US construction satisfies the simulatability property it is sufficient to require the scheme \mathcal{US} to be anonymous in the sense of [16]. A scheme \mathcal{US} is said to be *anonymous* if (roughly) for two randomly generated key pairs $(pk_0, sk_0), (pk_1, sk_1)$ and a message m , it is infeasible to distinguish the two distributions $\text{US.Sign}(sk_0, m)$ and $\text{US.Sign}(sk_1, m)$. More formally, we can prove the following theorem:

Theorem 3. *If \mathcal{S} is a secure standard signature scheme and \mathcal{US} is a secure anonymous undeniable signature scheme then IB_US as outlined above is a secure identity-based undeniable signature scheme.*

As far as we know, only one IB-US has been previously presented in [24]. This scheme uses bilinear pairings and it is proved secure in the random oracle model. We stress that the security model in [24] seems to be incomplete, as the authors do not consider simulatability.

In [16], an anonymous PKI-based US scheme based on the RSA primitive was proposed (the security proof uses the random oracle model). A different anonymous US scheme, whose security is proved in the standard model, can be found in [23]; it does not employ bilinear pairings, but the disavowal protocol is quite inefficient. Using these anonymous US schemes [16,23], we can obtain secure IB-US schemes in the random oracle model and also in the standard model, based on different computational assumptions, which do not employ bilinear pairings.

3.3 Aggregate Signatures

The idea of an aggregate signature scheme [7] is to combine n signatures on n different messages, signed by n (possibly different) signers, in order to obtain a single aggregate signature $AgSig$ which provides the same certainty than the

n initial signatures. The main goal in the design of such protocols is that the length of $AgSig$ be constant, independent of the number of messages and signers. Of course, to check correctness of an aggregate signature, the verifier will also need the messages m_i and the public keys pk_i , but this is not taken into account when considering the length of $AgSig$.

In the identity-based framework, the only proposal which achieves constant-length aggregation is that of [18]; however, this scheme only works in a more restrictive scenario where some interaction or sequentiality is needed among the signers of the messages which later will be aggregated (in the same direction as [25] for the PKI-based scenario). With respect to non-interactive aggregate signatures in the identity-based setting, the most efficient proposal is from [21], that does not achieve constant-length aggregation: the length of the aggregate signature does not depend on the number of signed messages, but on the number of different signers. Using the approach of this work, we can achieve exactly the same level of partial aggregation for identity-based signatures. In effect, let us consider our generic construction, and let us assume that the employed PKI-based signature scheme \mathcal{S} allows constant-length aggregation. The the input of the aggregation algorithm would be $\{(id_i, sig_{msk}(id_i||pk_i), pk_i, m_i, sig_i)\}_{1 \leq i \leq n}$, where sig_i and $sig_{sk_i}(m_i)$ are signatures resulting from scheme \mathcal{S} , and can therefore be aggregated into a PKI-based aggregate signature $AgSig$, of constant-length. Then the final identity-based aggregate signature would be $IBAgSig = (Ag_Sig, pk_1, \dots, pk_n)$. This aggregate signature, along with the n messages and the n identities, is sufficient to verify the correctness of the n signatures. Therefore the length of the identity-based aggregate signature $IBAgSig$ is linear with respect to the number of different signers.

3.4 Limitations and Extensions

Our generic approach to construct identity-based signature schemes with special properties does not work in situations where the signing procedure (in the corresponding PKI-based scheme) involves other public keys than the one from the signer, and interaction between the signer and the owners of these public keys is not mandatory. Our approach fails in this case because in the identity-based framework the signer only knows the identity of the other users, and needs some interaction with them in order to know the public key that they have received in the key extraction phase. Some examples of signature schemes with special properties falling inside this group are: ring signatures; designated verifier signatures; confirmer signatures; chameleon signatures; and nominative signatures.

We are aware of the fact that the list of properties where the generic approach can be applied is not complete and it obviously can also be applied to other concepts (like one-time signatures, homomorphic signatures, etc.) as well.

4 Generic Construction of ID-Based Blind Signatures

In this section we consider in more detail the generic construction in the case of blind signature schemes. In blind signature (BS) schemes [11] a user can ask

a signer to blindly sign a (secret) message m . At the end of the (interactive) signing process, the user obtains a valid signature on m , but the signer has no information about the message he has just signed. A formal security model of blind signatures was introduced in [22,27]. Partially blind signature schemes are a variation of this concept, where the signer can include some common information in the blind signature, under some agreement with the final receiver of the signature. This concept was introduced in [1] and the security of such schemes was formalized in [2].

The first identity-based blind signature (IB-BS) schemes were proposed in [31,30]. They employ bilinear pairings, but their security is not formally analyzed. Subsequent schemes were proposed in [13] but security is only provided in a weaker model (i.e. against sequential adversaries).

The main result of this section can be stated as follows.

Theorem 4. *If \mathcal{S} is a strongly secure standard signature scheme and \mathcal{PS} is a secure (partially) blind signature scheme then a secure identity-based (partially) blind signature scheme IB_PS can be constructed.*

Here the IB-BS scheme inherits the security properties of the BS scheme — if BS is secure against concurrent adversaries so is IB-BS. In particular, we obtain the first IB-BS scheme provably secure (in the standard model), against concurrent adversaries (by using the results from [8,26,15]), we obtain IB-BS schemes which do not employ bilinear pairings [4], and we obtain IB-BS schemes from any one-way trapdoor permutation [22,15].

We now formally prove Theorem 4. First we recall the basic definitions of PKI-based and identity-based blind signature schemes, then we explain and analyze our construction and prove its blindness. Due to lack of space, we included all details (definitions and analysis) related to the unforgeability property in the full version of this paper [17].

4.1 Blind Signature Schemes

Blind signature schemes were introduced in [11] with electronic banking as first motivation. The intuitive idea is that a user asks some signer to blindly sign a (secret) message m . At the end of the process, the user obtains a valid signature on m from the signer, but the signer has no information about the message he has signed. More formally, a blind signature scheme $\mathcal{BS} = (\text{BS.KG}, \text{BS.Sign}, \text{BS.Vfy})$ consists of the following (partially interactive) algorithms.

The **key generation** algorithm BS.KG takes as input a security parameter k and returns a secret key sk and a matching public key pk . We use notation $(sk, pk) \leftarrow \text{BS.KG}(1^k)$ to refer to one execution of this protocol. The **blind signing** algorithm BS.Sign is an interactive protocol between a user U and a signer S with public key pk . The input for the user is $\text{Inp}_U = (m, pk)$ where m is the message he wants to be signed by the signer. The input Inp_S of the signer is his secret key sk . In the end, the output Out_S of the signer is 'completed' or 'not completed', whereas the output Out_U of the user is either 'fail' or a signature $sig = sig_{sk}(m)$. We use notation $(\text{Out}_U, \text{Out}_S) \leftarrow \text{BS.Sign}(\text{Inp}_U, \text{Inp}_S)$ to refer

to one execution of this interactive protocol. Finally, the **verification** algorithm BS.Vfy is the same verification protocol as in standard signature schemes. To refer to one execution of this protocol, we use notation $\{0, 1\} \leftarrow \text{BS.Vfy}(m, sig)$.

BLINDNESS. Intuitively, the blindness property captures the notion of a signer who tries to obtain some information about the messages he is signing for some user. Formally, this notion is defined by the following game that an adversary (signer) \mathcal{B} plays against a challenger (who plays the role of a user).

First the adversary \mathcal{B} runs the key generation protocol $(sk, pk) \leftarrow \text{BS.KG}(1^k)$. Then the adversary \mathcal{B} chooses two messages m_0 and m_1 and sends them to the challenger, along with the public key pk . The challenger chooses $b \in \{0, 1\}$ at random and then the interactive signing protocol is executed two times (possibly in a concurrent way), resulting in $(Out_{U,b}, Out_{S,b}) \leftarrow \text{BS.Sign}(Inp_{U,b}, Inp_{S,b})$ and $(Out_{U,1-b}, Out_{S,1-b}) \leftarrow \text{BS.Sign}(Inp_{U,1-b}, Inp_{S,1-b})$, where adversary \mathcal{B} plays the role of the signer S , and the challenger plays the role of the user, with inputs $Inp_{U,b} = (pk, m_b)$ and $Inp_{U,1-b} = (pk, m_{1-b})$. Finally, the adversary \mathcal{B} outputs its guess b' . Note that the adversary in the above security game is in the possession of the secret key sk .

We say that such an adversary \mathcal{B} succeeds if $b' = b$ and define its advantage in the above game as $\text{Adv}_{\mathcal{BS}, \mathcal{B}}^{\text{blind}}(k) = |\Pr[b' = b] - 1/2|$. A scheme \mathcal{BS} has the blindness property if, for all PPT adversaries \mathcal{B} , $\text{Adv}_{\mathcal{BS}, \mathcal{B}}^{\text{blind}}(k)$ is a negligible function (with respect to the security parameter k).

4.2 Identity-Based Blind Signature Schemes

Analogously, an identity-based blind signature scheme $\text{IB-BS} = (\text{IB-BS.KG}, \text{IB-BS.Extr}, \text{IB-BS.Sign}, \text{IB-BS.Vfy})$ consists of the following algorithms.

The **setup** algorithm IB-BS.KG takes as input a security parameter k and returns, on the one hand, the master public key mpk and, on the other hand, the value master secret key msk , which is known only to the master entity. We note an execution of this protocol as $(msk, mpk) \leftarrow \text{IB-BS.KG}(1^k)$. The **key extraction** algorithm IB-BS.Extr takes as inputs mpk , the master secret key msk and an identity $id \in \{0, 1\}^*$, and returns a secret key $sk[id]$ for the user with this identity. We use notation $sk[id] \leftarrow \text{IB-BS.Extr}(msk, id)$ to refer to one execution of this protocol. The **blind signing** algorithm IB-BS.Sign is an interactive protocol between a user U and a signer with identity id . The common input for them is mpk . The input for the user is $Inp_U = (id, m)$ where m is the message he wants to be signed by id . The input Inp_{id} of the signer is his secret key $sk[id]$. In the end, the output Out_{id} of the signer is 'completed' or 'not completed', whereas the output Out_U of the user is either 'fail' or a signature $sig = sig_{msk}(id, m)$. We use notation $(Out_U, Out_{id}) \leftarrow \text{IB-BS.Sign}(mpk, Inp_U, Inp_{id})$ to refer to one execution of this interactive protocol. Finally, the **verification** algorithm IB-BS.Vfy takes as input mpk , a message m , an identity id and a signature sig ; it outputs 1 if the signature is valid with respect to the public key mpk and the identity id , and 0 otherwise. To refer to one execution of this protocol, we use notation $\{0, 1\} \leftarrow \text{IB-BS.Vfy}(mpk, id, m, sig)$.

An identity-based blind signature scheme must satisfy the requirements of correctness, blindness and unforgeability. Due to lack of space, we focus only on the blindness property.

BLINDNESS. Blindness of an identity-based blind signature scheme is defined by a game played between a challenger and an adversary. This adversary \mathcal{B}_{IB} models the dishonest behavior of a signer who tries to distinguish which message (between two messages chosen by himself) is being signed in an interactive execution of the signing protocol with a user. The game is as follows.

First the challenger runs the setup protocol $(msk, mpk) \leftarrow \text{IB_BS.KG}(1^k)$ and gives mpk to \mathcal{B}_{IB} . The master secret key msk is kept secret by the challenger. The adversary \mathcal{B}_{IB} is allowed to query for secret keys of identities id_i of his choice. The challenger runs $sk[id_i] \leftarrow \text{IB_BS.Extr}(msk, id_i)$ and gives the resulting secret key $sk[id_i]$ to \mathcal{B}_{IB} . If the same identity is asked again, the same value $sk[id_i]$ must be returned by the challenger. At some point, the adversary \mathcal{B}_{IB} chooses an identity id^* and two messages m_0, m_1 , and sends these values to the challenger. The challenger chooses $b \in \{0, 1\}$ at random and then the interactive signing protocol is executed twice (possibly in a concurrent way), resulting in $(Out_{U,b}, Out_{id^*,b}) \leftarrow \text{IB_BS.Sign}(Inp_{U,b}, Inp_{id^*,b})$ and $(Out_{U,1-b}, Out_{id^*,1-b}) \leftarrow \text{IB_BS.Sign}(Inp_{U,1-b}, Inp_{id^*,1-b})$, where adversary \mathcal{B}_{IB} plays the role of the signer id^* , and the challenger plays the role of the user, with inputs $Inp_{U,b} = (m_b, id^*)$ and $Inp_{U,1-b} = (m_{1-b}, id^*)$. Finally, the adversary \mathcal{B}_{IB} outputs its guess b' .

We say that such an adversary \mathcal{B} succeeds if $b' = b$ and define its advantage in the above game as $\text{Adv}_{\text{IB_BS}, \mathcal{B}_{\text{IB}}}^{\text{ib-blind}}(k) = |\Pr[b' = b] - 1/2|$. A scheme IB_BS has the blindness property if, for all PPT adversaries \mathcal{B}_{IB} , $\text{Adv}_{\text{IB_BS}, \mathcal{B}_{\text{IB}}}^{\text{ib-blind}}(k)$ is a negligible function (with respect to the security parameter k).

4.3 Constructing Identity-Based Blind Signature Schemes

Let $\mathcal{S} = (\text{S.KG}, \text{S.Sign}, \text{S.Vfy})$ be a standard signature scheme and let $\mathcal{BS} = (\text{BS.KG}, \text{BS.Sign}, \text{BS.Vfy})$ be a blind signature scheme. We construct an ID-based blind signature scheme $\text{IB_BS} = (\text{IB_BS.KG}, \text{IB_BS.Sign}, \text{IB_BS.Extr}, \text{IB_BS.Vfy})$ as follows.

SETUP $\text{IB_BS.KG}(1^k)$: On input a security parameter k , the key generation protocol S.KG of \mathcal{S} is executed, resulting in $(SK, PK) \leftarrow \text{S.KG}(1^k)$. The master public key is defined as $mpk = PK$, whereas the master secret key stored by the master entity is $msk = SK$.

KEY EXTRACTION $\text{IB_BS.Extr}(msk, id_i)$: When the user secret key $sk[id_i]$ for some identity id_i is requested, the master entity first checks if it already has established a user secret key for id_i . If so, the old secret key is returned. Otherwise it generates and stores a new user secret key as follows: it runs the key generation protocol of the blind signature scheme \mathcal{BS} , resulting in $(sk_i, pk_i) \leftarrow \text{BS.KG}(1^k)$. Then it uses signature scheme \mathcal{S} to sign the "message" $id_i \parallel pk_i$, that is, it executes $sig_{msk}(id_i \parallel pk_i) \leftarrow \text{S.Sign}(msk, id_i \parallel pk_i)$. The resulting secret key, which is sent to the owner of the identity, is $sk[id_i] = (sk_i, pk_i, sig_{msk}(id_i \parallel$

pk_i)). The recipient can verify the obtained secret key by executing $\{0, 1\} \leftarrow \text{S.Vfy}(mpk, id_i \parallel pk_i, sig_{msk}(id_i \parallel pk_i))$; if the output is 1, then the secret key is accepted.

BLIND SIGNATURE IB_BS .Sign: The interactive protocol between a user U and a signer with identity id_i consists of the following steps (recall that mpk is a common input for user and signer, the input of the user is (id_i, m) and the input of the signer is $sk[id_i]$).

1. User U sends the query $(id_i, \text{'blindsignature?'})$ to the signer.
2. If the signer does not want to sign, the protocol finishes with $Out_U = \text{'fail'}$ and $Out_{id_i} = \text{'not completed'}$. Otherwise, the signer sends $(pk_i, sig_{msk}(id_i \parallel pk_i))$ back to the user.
3. The user runs $\{0, 1\} \leftarrow \text{S.Vfy}(mpk, id_i \parallel pk_i, sig_{msk}(id_i \parallel pk_i))$. If the output is 0, then the protocol finishes with $Out_U = \text{'fail'}$ and $Out_{id_i} = \text{'not completed'}$. Otherwise, user and signer interact to run the blind signature protocol of \mathcal{BS} , resulting in $(Out'_U, Out'_{id_i}) \leftarrow \text{BS.Sign}(Inp_U, Inp_{id_i})$, where $Inp_U = (pk_i, m)$ and $Inp_{id_i} = sk_i$. If $Out'_U \neq \text{'fail'}$, then it consists of a standard signature $sig_{sk_i}(m)$ on m under secret key sk_i . The final output for the user is in this case $Out_U = sig(id_i, m_i) = (sig_{msk}(id_i \parallel pk_i), pk_i, sig_{sk_i}(m))$, which is defined to be the identity-based signature on message m from identity id_i .

VERIFICATION IB_BS .Vfy($mpk, id_i, m, sig(id_i, m_i)$): Given as input a message m , an identity id_i and an identity-based signature $sig(id_i, m_i)$ that is parsed as $(sig_{msk}(id_i \parallel pk_i), pk_i, sig_{sk_i}(m))$, the verification protocol works as follows. The two verification protocols, of schemes \mathcal{S} and \mathcal{BS} , are executed in parallel: $\{0, 1\} \leftarrow \text{S.Vfy}(mpk, id_i \parallel pk_i, sig_{msk}(id_i \parallel pk_i))$ and $\{0, 1\} \leftarrow \text{BS.Vfy}(pk_i, m, sig_{sk_i}(m))$. If both outputs are 1, then the final output of this protocol is also 1. Otherwise, the output is 0.

4.4 Security Analysis

In this section we prove that the identity-based blind signature scheme IB_BS constructed in the previous section satisfies the blindness property, assuming that the schemes \mathcal{S} and \mathcal{BS} employed as primitives are secure. The detailed analysis of the unforgeability property can be found in [17].

Theorem 5. *Assume the signature scheme \mathcal{S} is strongly unforgeable and the blind signature scheme \mathcal{BS} is blind. Then the identity-based blind signature scheme IB_BS constructed in Section 4.3 is blind.*

Proof. Assume there exists a successful adversary \mathcal{B}_{IB} against the blindness of the scheme IB_BS . We show that then there exists either a successful forger \mathcal{F} against the signature scheme \mathcal{S} or a successful adversary \mathcal{B} against the blindness of the blind signature scheme \mathcal{BS} . We now construct \mathcal{F} and \mathcal{B} .

Setup. Forger \mathcal{F} receives as initial input some public key PK for the standard signature scheme \mathcal{S} . Then we initialize the adversary \mathcal{B}_{IB} by providing it with $mpk = PK$.

Secret key queries. Adversary \mathcal{B}_{IB} is allowed to make secret key queries for identities id_i of its choice. To answer a query, we run the key generation protocol of the blind signature scheme \mathcal{BS} to obtain $(sk_i, pk_i) \leftarrow \text{BS.KG}(1^k)$. Then we send the query $m_i = id_i \parallel pk_i$ to the signing oracle of the forger \mathcal{F} , and obtain as answer a valid signature sig_i with respect to scheme \mathcal{S} and public key $PK = mpk$. Then we send to \mathcal{B}_{IB} the consistent answer $sk[id_i] = (sk_i, pk_i, sig_i)$. We store all this information in a table. If the same identity is asked twice by \mathcal{B}_{IB} , then the same secret key is given as answer.

Challenge. At some point, \mathcal{B}_{IB} will output some challenge identity id_* and two messages m_0, m_1 . Without loss of generality we can assume that \mathcal{B}_{IB} had already asked for the secret key of this identity (otherwise, we generate it now and send it to \mathcal{B}_{IB}), obtaining $sk[id_*] = (sk_*, pk_*, sig_*)$. Then we start constructing an adversary \mathcal{B} against the blindness of the scheme \mathcal{BS} , by sending public key pk_* and messages m_0, m_1 to the corresponding challenger. Now we must execute twice the interactive blind signature protocol with \mathcal{B}_{IB} , where \mathcal{B}_{IB} acts as a signer and we act as the user. For both executions, we first send $(id_*, \text{'blindsignature?'})$ to \mathcal{B}_{IB} . As answers, we will obtain $(pk_*^{(0)}, sig_*^{(0)})$ and $(pk_*^{(1)}, sig_*^{(1)})$ from \mathcal{B}_{IB} , where $sig_*^{(j)}$ is a valid signature on $id_* \parallel pk_*^{(j)}$, for both $j = 0, 1$.

If $(pk_*^{(j)}, sig_*^{(j)}) \neq (pk_*, sig_*)$ for either $j = 0$ or $j = 1$, then \mathcal{F} outputs $sig_*^{(j)}$ as a valid forgery on the message $id_* \parallel pk_*^{(j)}$ for the signature scheme \mathcal{S} . This is a valid forgery against signature scheme \mathcal{S} , because these signatures were not obtained during the attack. Therefore, in this case we would have a successful forger \mathcal{F} against \mathcal{S} , contradicting the hypothesis in the statement of the theorem which claims that \mathcal{S} is strongly unforgeable.

From now on we assume $(pk_*^{(j)}, sig_*^{(j)}) = (pk_*, sig_*)$ for both $j = 0, 1$ and the two first steps in the two executions of the interactive signing protocol are identical. Then we run the two execution of the blind signing protocol of scheme \mathcal{BS} , playing the role of the signer: we obtain from \mathcal{B}_{IB} the information that we must send to the challenger (user) of \mathcal{BS} , and this challenger sends back to us the information that we must provide to \mathcal{B}_{IB} . This challenger of \mathcal{BS} is the one who chooses the bit $b \in \{0, 1\}$.

Eventually, adversary \mathcal{B}_{IB} outputs its guess b' . \mathcal{B} outputs the same bit b' as its guess in the blind game against the blind signature scheme \mathcal{BS} .

The first two steps in the two executions of the interactive signing protocol of IB_BS run between \mathcal{B}_{IB} and us are identical. Hence distinguishing between the two executions of IB_BS.Sign is equivalent to distinguishing between the two executions of BS.Sign . This completes the proof. \square

We stress that the signature scheme \mathcal{S} really has to be *strongly* unforgeable; otherwise a signer can break blindness by using different versions of $sk[id_i]$ in different signing sessions and later use this information to trace the user.

Theorem 6. *Assume the standard signature scheme S is unforgeable and the blind signature scheme \mathcal{BS} is unforgeable. Then the identity-based blind signature scheme IB_BS from Section 4.3 is unforgeable.*

The proof of Theorem 6 can be found in [17]. Theorems 5 and 6 imply Theorem 4.

Acknowledgements

We thank the anonymous referees for their comments that helped improving the presentation of our results. The work of the second author has been carried out during the tenure of an ERCIM Fellowship (<http://www.ercim.org/activity/fellows/>). The third author was supported by the research program Sentinels (<http://www.sentinels.nl>). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

References

1. M. Abe and E. Fujisaki. How to date blind signatures. *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251, 1996.
2. M. Abe and T. Okamoto. Provably secure partially blind signatures. *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286, 2000.
3. M. Bellare, C. Namprempre and G. Neven. Security proofs for identity-based identification and signature schemes. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 268–286, 2004.
4. M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
5. M. Bellare and P. Rogaway. Optimal asymmetric encryption. *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111, 1994.
6. D. Boneh and M.K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
7. D. Boneh, C. Gentry, B. Lynn and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432, 2003.
8. J. Camenisch, M. Koprowski and B. Warinschi. Efficient blind signatures without random oracles. *SCN 04*, volume 3352 of *LNCS*, pages 134–148, 2004.
9. R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. *30th ACM STOC*, pages 209–218, 1998.
10. J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. *PKC 2003*, volume 2567 of *LNCS*, pages 18–30, 2003.
11. D. Chaum. Blind signatures for untraceable payments. *CRYPTO'82*, pages 199–203, USA, 1983.
12. D. Chaum and H. Van Antwerpen. Undeniable signatures. *CRYPTO'89*, volume 435 of *LNCS*, pages 212–216, 1990.
13. S.M. Chow, L.K. Hui, S.M. Yiu and K.P. Chow. Two improved partially blind signature schemes from bilinear pairings. *ACISP 2005*, pages 316–325, 2005.

14. I. Damgard and T.P. Pedersen. New convertible undeniable signature schemes. *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 372–386, 1996.
15. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77, 2006.
16. S.D. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. *CT-RSA 2003*, volume 2612 of *LNCS*, pages 80–97, 2003.
17. D. Galindo, J. Herranz. and E. Kiltz. On the generic construction of identity-based signatures with additional properties. Cryptology ePrint Archive, Report 2006/296, 2006. Full version of this paper, <http://eprint.iacr.org/>.
18. C. Gentry and Z. Ramzan. Identity-based aggregate signatures. *PKC 2006*, volume 3958 of *LNCS*, pages 257–273, 2006.
19. S. Goldwasser, S. Micali and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
20. C. Gu and Y. Zhu. An id-based verifiable encrypted signature scheme based on Hess's scheme. *CISC'05*, pages 42–52, 2005.
21. J. Herranz. Deterministic identity-based signatures for partial aggregation. *The Computer Journal*, 49 (3):322–330, 2006.
22. A. Juels, M. Luby and R. Ostrovsky. Security of blind digital signatures (extended abstract). *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164, 1997.
23. F. Laguillaumie and D. Vergnaud. Short undeniable signatures without random oracles: the missing link. *Indocrypt'05*, volume 3797 of *LNCS*, pages 283–296, 2005.
24. B. Libert and J.J. Quisquater. Identity based undeniable signatures. *CT-RSA 2004*, volume 2964 of *LNCS*, pages 112–125, 2004.
25. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. *EUROCRYPT'06*, 2006.
26. T. Okamoto. Efficient blind and partially blind signatures without random oracles. *TCC 2006*, volume 3876 of *LNCS*, pages 80–99, 2006.
27. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
28. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing over elliptic curve (in Japanese). *SCIS 2001*, Jan 2001.
29. A. Shamir. Identity-based cryptosystems and signature schemes. *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53, 1985.
30. F. Zhang and K. Kim. Efficient ID-based blind signature and proxy signature from bilinear pairings. In *ACISP'03*, pages 312–323, 2003.
31. F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 533–547, 2002.

On the Provable Security of an Efficient RSA-Based Pseudorandom Generator

Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang

Centre for Advanced Computing – Algorithms and Cryptography (ACAC)
Dept. of Computing, Macquarie University, North Ryde, Australia
{rons, josef, hwang}@comp.mq.edu.au
<http://www.ics.mq.edu.au/acac/>

Abstract. Pseudorandom Generators (PRGs) based on the RSA inversion (one-wayness) problem have been extensively studied in the literature over the last 25 years. These generators have the attractive feature of provable pseudorandomness security assuming the hardness of the RSA inversion problem. However, despite extensive study, the most efficient provably secure RSA-based generators output asymptotically only at most $O(\log n)$ bits per multiply modulo an RSA modulus of bitlength n , and hence are too slow to be used in many practical applications.

To bring theory closer to practice, we present a simple modification to the proof of security by Fischlin and Schnorr of an RSA-based PRG, which shows that one can obtain an RSA-based PRG which outputs $\Omega(n)$ bits per multiply and has provable pseudorandomness security assuming the hardness of a well-studied variant of the RSA inversion problem, where a constant fraction of the plaintext bits are given. Our result gives a positive answer to an open question posed by Gennaro (J. of Cryptology, 2005) regarding finding a PRG beating the rate $O(\log n)$ bits per multiply at the cost of a reasonable assumption on RSA inversion.

Keywords: Pseudorandom generator, RSA, provable security, lattice attack.

1 Introduction

Background. The RSA Pseudorandom bit generator (RSA PRG) works by iterating the RSA encryption mapping $x \rightarrow x^e \bmod N$ (with public RSA modulus N of length n bits and public exponent e coprime to $\phi(N)$) on a secret random initial seed value $x_0 \in \mathbb{Z}_N$ to compute the intermediate state values $x_{i+1} = x_i^e \bmod N$ (for $i = 0, 1, 2, \dots$) and outputting r least-significant bits of the state value x_i per iteration. The pseudorandomness of the RSA PRG (especially the case $r = 1$) was studied extensively by several researchers [19,2,30,1,14]. However, even the best security proof so far [14,28] only applies to the case when only a very small number of bits $r = O(\log n)$ is output per iteration. Consequently, even with small public exponent e , these proven RSA PRG variants only output $O(\log n)$ bits per multiply modulo N and hence are too slow for most practical applications. As far as we are aware, these are currently the most efficient RSA-based PRGs with proven pseudorandomness security.

Our Approach. Our approach to studying the provable security of efficient variants of the RSA PRG is based on two observations.

First, we observe that existing security proofs of the RSA PRG have always attempted to prove the security assuming the hardness of the classical RSA one-wayness problem (given RSA modulus N and $y = x^e \bmod N$ for random $x \in \mathbb{Z}_N$, find x). If we instead make a stronger hardness assumption, we can hope to prove the security of much more efficient and practical variants of the RSA PRG, with $r = \Omega(n)$. But we must be careful in choosing this stronger hardness assumption to ensure that it is based on substantial evidence – it must be a hard problem which has been undoubtedly studied extensively by experts. This leads to our second observation.

Our second observation is that over the last decade, beginning with the work of Coppersmith [11], the following variant of the RSA one-wayness problem has been studied explicitly:

(δ, e) -Small Solution RSA ((δ, e) -SSRSA) Problem. Given a random n -bit RSA modulus N , the coefficients of a univariate polynomial $f(z) = a_e z^e + a_{e-1} z^{e-1} + \dots + a_0 \in \mathbb{Z}_N[z]$ of degree e (with $a_e \in \mathbb{Z}_N^*$) and $y = f(\bar{z}) \bmod N$ for a random integer $\bar{z} < N^\delta$ (with $0 < \delta < 1$), find \bar{z} (note that we will only be interested in instances where f is such that \bar{z} is uniquely determined by (N, f, y)).

The celebrated lattice-based attack of Coppersmith [11] shows that for small e , the (δ, e) -SSRSA problem can be solved in polynomial time (in n) whenever $\delta < 1/e$. But when $\delta > 1/e + \epsilon$ for some constant $\epsilon > 0$, the lattice attack fails, and the only known attack (beyond factoring N) is to run the lattice attack $O(N^\epsilon)$ times for each guess of the $\epsilon \cdot n$ most-significant bits of \bar{z} . Hence, when ϵ is made sufficiently large to make the above lattice attack slower than factoring N (namely even $\epsilon = O((\log n/n)^{2/3})$ suffices), the best known attack against $(1/e + \epsilon, e)$ -SSRSA problem is to factor N . Importantly, this hardness assumption is supported by explicit evidence in the literature that the $(1/e + \epsilon, e)$ -SSRSA problem has been studied by experts [12,26,10], yet these studies have not yielded an efficient algorithm for the $(1/e + \epsilon, e)$ -SSRSA problem.

Our Result. We present a simple modification to the proof of security of the RSA PRG by Fischlin and Schnorr [14] which shows that assuming the hardness of a certain specific $(1/e + \epsilon, e)$ -SSRSA one-wayness problem suffices to prove the pseudorandomness of the RSA PRG outputting $r = (1/2 - 1/e - \epsilon - o(1)) \cdot n$ LS bits per iteration. Our specific $(1/e + \epsilon, e)$ -SSRSA one-wayness problem can be posed as RSA inversion with some known plaintext bits, namely: Given N , $y = [x^e]_N$, r LS bits of x and $w \approx n/2$ MS bits of x , for $x \in_R \mathbb{Z}_N$, find x . For small (constant) $e \geq 3$ we therefore obtain a throughput of $\Omega(n)$ output pseudorandom bits per multiply modulo the RSA modulus N , which is a significant improvement over the $O(\log n)$ bits per multiply throughput obtained using previous proof of security relative to the RSA assumption. We believe this answers in the positive an open question raised by Gennaro [15], who asked whether one can obtain a PRG which beats the rate $O(\log n)$ bits per multiply at the cost of a stronger but reasonable assumption on RSA inversion.

Organization. In Section 1.1 we discuss additional related work. Section 2 contains definitions and notations. In Section 3, we review the RSA PRG construction and its proof of security by Fischlin and Schnorr [14]. Section 4 presents our modified security proof for the RSA PRG assuming the hardness of a $(1/e + \epsilon, e)$ -SSRSA problem. In Section 5, we estimate concrete parameters and associated PRG performance for given proven security level and security assumptions. In Section 6 we investigate the potential for performance improvements using a stronger hardness assumption. Section 7 concludes the paper with some open problems.

1.1 Additional Related Work

Related PRG constructions can be divided in two classes.

The first class contains PRGs based on related hardness assumptions. The well known Blum-Blum-Shub (BBS) generator [6] has the same structure as the RSA PRG, but uses the Rabin squaring iteration function instead. Similar security results as for the RSA PRG are known for this generator [14], but we need a less known assumption to prove the security of efficient variants of this generator (see Section 6). The factoring-based construction by Goldreich and Rosen [17] has a throughput of $O(1)$ bits per multiply modulo an n bit modulus. The Miceli-Schnorr RSA-based constructions [24] have a throughput of $\Omega(n)$ bits per multiply, but their pseudorandomness security is only proven assuming the *pseudorandomness* of the RSA function with small inputs whereas for our construction we can prove pseudorandomness assuming only a much weaker assumption of *one-wayness* of RSA with small inputs. The PRG of Boneh et al [9] also achieves a throughput of $\Omega(n)$ bits per multiply (and in fact may use a smaller *prime* modulus), but its provable pseudorandomness security also relies on a pseudorandomness assumption rather than a one-wayness assumption.

The second class of PRGs achieve provable pseudorandomness based on different one-wayness assumptions. The construction by Impagliazzo and Naor [21] is based on the hardness of the Subset Sum problem. Although this construction is potentially very efficient, its concrete security against lattice-based subset sum attacks is difficult to estimate and requires carefully chosen large parameters with a small number of bits output per function evaluation. Very recently, a more practical ‘QUAD’ construction by Berbain et al [3] was proposed, using similar ideas to [21] in its security proof, but based on the hardness of solving a random system of multivariate quadratic equations over a finite field (‘MQ’ problem). We compare the practical performance of our construction with QUAD in Section 5. The fastest PRG based on the hardness of a variant of the Discrete-Log one-wayness problem is due to Gennaro [15] (improving on earlier work by Patel and Sundaram [27]), but its throughput is at most $O((\frac{n}{\log n})^{2/3}) = o(n)$ bits per multiply, compared to $\Omega(n)$ bits per multiply for our construction with same modulus length n and conjectured security level.

Finally, we also wish to mention the lattice-based attacks of Blackburn et al [5,4] on a class of PRGs having the same iterative structure as our RSA PRG. These attacks show that the RSA PRG is insecure when the number of bits output per iteration r is larger than about $\frac{2}{3}n$ [5] for $e = 2$, and about

$(1 - \frac{1}{e^{(e+1)/2+2}})n$ [4] in the general case (these results are obtained for r MS bits output per iteration and prime moduli, but we believe that with appropriate modifications they hold also for r LS bits and RSA moduli). We remark that the general case attacks in [4] use low-dimension lattices and are rigorously proven. A heuristic extension of these attacks to high dimension lattices using the Coppersmith method [11] suggests that the RSA PRG is insecure asymptotically with $r \geq (1 - \frac{1}{e+1})n$ (we omit details of these calculations here). These lower bounds for insecure values of r are greater by a factor of about 2 than the upper bounds on r for which our security proof applies. Closing this remaining gap between best attack and best proof is an interesting open problem.

2 Preliminaries

Notation. For integers x and N , we use $[x]_N$ to denote the remainder $x \bmod N$. We use $L_r(x) = [x]_{2^r}$ to denote the r least significant bits of the binary representation of x . Similarly, we use $M_r(x) = (x - L_{n-r}(x))/2^{n-r}$ (where n is the bit length of x) to denote the r most significant bits of the binary representation of x . For $x \in \mathbb{Z}_N$, we use $\widehat{M}_{N,r}(x)$ to denote any approximation of x with additive error $|x - \widehat{M}_{N,r}(x)| \leq N/2^r$.

Probability Distributions and Distinguishers. Let \mathcal{D} denote a probability distribution over $\{0, 1\}^\ell$. We denote by $s \leftarrow \mathcal{D}$ the assignment to s of a random element sampled from the distribution \mathcal{D} . If S denotes a set then we let $s \in_R S$ denote the assignment to s of a *uniformly* random element sampled from S . Let \mathcal{D}_1 and \mathcal{D}_2 denote two probability distributions on some finite set. We say that an algorithm D is a (T, δ) distinguisher between \mathcal{D}_1 and \mathcal{D}_2 if D runs in time at most T and has distinguishing advantage at least δ between \mathcal{D}_1 and \mathcal{D}_2 , i.e. $|\Pr_{s \leftarrow \mathcal{D}_1}[D(s) = 1] - \Pr_{s \leftarrow \mathcal{D}_2}[D(s) = 1]| \geq \delta$. The *statistical distance* between two distributions \mathcal{D}_1 and \mathcal{D}_2 is $\frac{1}{2} \sum_s |\mathcal{D}_1(s) - \mathcal{D}_2(s)|$. It gives an upper bound on the distinguishing advantage of any distinguisher between \mathcal{D}_1 and \mathcal{D}_2 , regardless of run-time.

Pseudorandom Bit Generators (PRGs). We use the following definition of pseudorandom generators and their concrete pseudorandomness.

Definition 1 ((T, δ) PRG). A (T, δ) Pseudorandom Generator (family) PRG is a collection of functions $G_N : \mathcal{S}_N \rightarrow \{0, 1\}^\ell$ indexed by $N \in \mathcal{I}_n$. Here \mathcal{I}_n (PRG function index space) and \mathcal{S}_N (PRG seed domain) are both efficiently samplable subsets of $\{0, 1\}^n$, where n is the security parameter. We require that any (probabilistic) distinguisher algorithm D running in time T has distinguishing advantage at most δ between the pseudorandom distribution $\mathcal{D}_{P,\ell}$ and the random distribution $\mathcal{D}_{R,\ell}$ on ℓ -bit strings, which are defined as follows:

$$\mathcal{D}_{P,\ell} = \{s : N \in_R \mathcal{I}_n; x_0 \in_R \mathcal{S}_N; s = G_N(x_0)\}$$

while

$$\mathcal{D}_{R,\ell} = \{s : s \in_R \{0, 1\}^\ell\}.$$

If algorithm D runs in time T and has distinguishing advantage at least δ between $\mathcal{D}_{P,\ell}$ and $\mathcal{D}_{R,\ell}$, we say that D is a (T, δ) distinguisher for PRG.

The RSA Inversion Problem. The classical RSA inversion problem is defined as follows.

Definition 2 ((n, e)-RSA problem). Let e be a fixed integer. Let \mathcal{I}_n denote the set of all n -bit RSA moduli $N = pq$ (for p, q primes of $n/2$ bits each) such that $\gcd(e, (p-1)(q-1)) = 1$. The (n, e) -RSA inversion problem is the following: given $N \in_R \mathcal{I}_n$ and $y = [x^e]_N$ for $x \in_R \mathbb{Z}_N$, find x . We say that algorithm A is a (T, ϵ) inversion algorithm for (n, e) -RSA if A runs in time T and succeeds with probability ϵ over the choice of $N \in_R \mathcal{I}_n$, $x \in_R \mathbb{Z}_N$ and the random coins of A .

Lattices. Let $\{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ be a set of n linearly independent vectors in \mathbb{R}^n . The set

$$\mathcal{L} = \{\mathbf{z} : \mathbf{z} = c_1 \mathbf{b}_1 + \dots + c_n \mathbf{b}_n; c_1, \dots, c_n \in \mathbb{Z}\}$$

is called an n -dimensional (full-rank) lattice with basis $\{\mathbf{b}_1, \dots, \mathbf{b}_n\}$. Given a basis $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ for a lattice \mathcal{L} , we define the associated basis matrix $M_{\mathcal{L}, \mathbf{B}}$ to be the (full-rank) $n \times n$ matrix whose i th row is the i th basis vector \mathbf{b}_i for $i = 1, \dots, n$. The quantity $|\det(M_{\mathcal{L}, \mathbf{B}})|$ is independent of \mathbf{B} . It is called the determinant of the lattice \mathcal{L} and denoted by $\det(\mathcal{L})$. Given any basis of a lattice \mathcal{L} , the well-known LLL algorithm [22] outputs in polynomial time a reduced basis for \mathcal{L} consisting of short vectors. We use the following result [8] bounding the length of those vectors.

Lemma 1. Let \mathcal{L} be a lattice of dimension d with basis matrix $\mathbf{B}_{\mathcal{L}}$ in lower diagonal form whose diagonal elements are greater or equal to 1. Then the Euclidean norm of the first two vectors in the LLL reduced basis for \mathcal{L} is at most $2^{d/2}(\det(\mathcal{L}))^{\frac{1}{d-1}}$.

3 Overview of the Fischlin-Schnorr Security Proof

The RSA PRG. We begin by recalling the RSA PRG construction.

Definition 3 ((n, e, r, ℓ)-RSAPRG Pseudorandom Generator). The pseudorandom generator family (n, e, r, ℓ) -RSAPRG is defined as follows. The PRG function index space \mathcal{I}_n is the set of all n -bit RSA moduli $N = pq$ (for p, q primes of $n/2$ bits each) such that $\gcd(e, (p-1)(q-1)) = 1$. Given index $N \in \mathcal{I}_n$ the PRG seed domain is \mathbb{Z}_N . Assume that ℓ is a multiple of r . Given a seed $x_0 \in_R \mathbb{Z}_N$, the PRG function $G_N : \mathbb{Z}_N \rightarrow \{0, 1\}^\ell$ is defined by

$$G_N(x_0) = (s_0, \dots, s_{\ell/r-1}) : s_i = L_r(x_i), x_{i+1} = [x_i^e]_N \text{ for } i = 0, \dots, \ell/r - 1.$$

As will become clear below, our result builds on the Fischlin-Schnorr result in essentially a ‘black box’ way, so our result can be understood without knowing most of the internal details of the reduction in [14]. Hence, in this section we provide only a very high-level overview of the basic security reduction [14] for

the RSA PRG from the RSA assumption, in the case of r LS bits output per iteration (refer to the full version of the paper [29] for more details).

Using our notation, the Fischlin-Schnorr security result can be stated concretely as follows.

Theorem 1 (Fischlin-Schnorr [14]). *For all $n \geq 2^9$, any (T, δ) distinguisher D for (n, e, r, ℓ) -RSAPRG can be converted into a $(T_{INV}, \delta/9)$ inversion algorithm A for the (n, e) -RSA problem with run-time at most*

$$T_{INV} = 2^{2r+14}(\ell/\delta)^6 n \log(n) \cdot (T + O(\ell/r \log(e)n^2)). \tag{1}$$

Proof. We are given a distinguisher D with run-time T and distinguishing advantage $\text{Adv}(D) \geq \delta$ between the *pseudorandom distribution* $\mathcal{D}_{P,\ell}$ (obtained by iterating $m = \ell/r$ times and outputting r LS bits per iteration) and the *random distribution* $\mathcal{D}_{R,\ell}$ on ℓ bit strings, namely:

$$\mathcal{D}_{P,\ell} = \{G_N(x_0) : N \in_R \mathcal{I}_n; x_0 \in_R \mathbb{Z}_N\}$$

while

$$\mathcal{D}_{R,\ell} = \{s : s \in_R \{0, 1\}^\ell\}.$$

We use D to construct the (n, e) -RSA inversion algorithm A as follows.

As a first step, we note that the pseudorandom distribution $\mathcal{D}_{P,\ell}$ is taken over the random choice of modulus $N \in_R \mathcal{I}_n$ as well as random seed $x_0 \in_R \mathbb{Z}_N$. For the remainder of the proof, we wish to fix N and find a lower bound on the distinguishing advantage $\text{Adv}_N(D)$ between $\mathcal{D}_{R,\ell}$ and the pseudorandom distribution $\mathcal{D}_{P,\ell,N}$ taken over just the random choice of $x_0 \in_R \mathbb{Z}_N$ for this fixed N , that is:

$$\mathcal{D}_{P,\ell,N} = \{G_N(x_0) : x_0 \in_R \mathbb{Z}_N\}.$$

To do so, we use an averaging argument over N .

Lemma 2. *There exists a subset $\mathcal{G}_n \subseteq \mathcal{I}_n$ of size at least $|\mathcal{G}_n| \geq \delta/2|\mathcal{I}_n|$ such that D has distinguishing advantage at least $\delta/2$ between the distributions $\mathcal{D}_{P,\ell,N}$ and $\mathcal{D}_{R,\ell}$ for all $N \in \mathcal{G}_n$.*

From now on we assume that $N \in \mathcal{G}_n$ (which happens with probability at least $\delta/2$ over $N \in_R \mathcal{I}_n$) so that D has distinguishing advantage at least $\delta/2$ between $\mathcal{D}_{P,\ell,N}$ and $\mathcal{D}_{R,\ell}$ (We remark that this first step is actually omitted in [14] which always assumes a fixed N ; however we add this step since we believe it is essential for a meaningful security proof: to demonstrate an efficient algorithm for RSA inversion contradicting the RSA assumption, one must evaluate its success probability over the random choice of modulus N , since for any fixed N an efficient algorithm always exists; it has built into it the prime factors of N).

We now convert ℓ/r -iteration distinguisher D into a 1-iteration distinguisher D' . This is a ‘hybrid’ argument using the fact that the mapping $x \rightarrow [x^e]_N$ is a permutation on \mathbb{Z}_N . Note that the ‘hybrid’ argument underlying this reduction has been known since the work of [18,7] and it is not explicitly included in [14].

Lemma 3 (*$m = \ell/r$ iterations to 1 iteration.*). Any (T, δ) distinguisher D between the m -iteration pseudorandom distribution $\mathcal{D}_{P,\ell,N}$ and the random distribution $\mathcal{D}_{R,\ell}$ can be converted into a $(T + O(m \log(e)n^2), \delta/m)$ 1-iteration distinguisher D' between the distributions

$$\mathcal{D}'_{P,r,N} = \{(y = [x^e]_N, s = L_r(x)) : x \in_R \mathbb{Z}_N\}$$

and

$$\mathcal{D}'_{R,r,N} = \{(y = [x^e]_N, s) : x \in_R \mathbb{Z}_N; s \in_R \{0, 1\}^r\}.$$

The main part of the Fischlin-Schnorr reduction [14] is the conversion of the distinguisher D' into an inversion algorithm that recovers the RSA preimage x from $y = [x^e]_N$ with the help of some additional information on x , namely r least-significant bits of $[ax]_N$ and $[bx]_N$ for some randomly chosen known $a, b \in \mathbb{Z}_N$, as well as rough approximations to $[ax]_N$ and $[bx]_N$. This is stated more precisely as follows.

Lemma 4 (Distinguisher to Inverter). For all $n \geq 2^9$, any (T, δ) distinguisher D' between the distributions $\mathcal{D}'_{P,r,N}$ and $\mathcal{D}'_{R,r,N}$ (see Lemma 3) can be converted into an inversion algorithm A' that, given N and $(y = [x^e]_N, a \in_R \mathbb{Z}_N, s_1 = L_r([ax]_N), u_1 = \widehat{M}_{N,k}([ax]_N), b \in_R \mathbb{Z}_N, s_2 = L_r([bx]_N), u_2 = \widehat{M}_{N,l}([bx]_N))$, for any $x \in \mathbb{Z}_N$ with $k = 3 \log(r/\delta) + 4$ and $l = \log(r/\delta) + 4$, outputs x with probability $\epsilon'_{INV} \geq 2/9$ (over the choice of $a \in_R \mathbb{Z}_N, b \in_R \mathbb{Z}_N$ and the random coins of A') and runs in time $T'_{INV} = 4n \log(n)(r/\delta)^2 \cdot (T + O(n^2))$. Here $\widehat{M}_{N,k}(x)$ denotes any approximation of x with additive error $|\widehat{M}_{N,k}(x) - x| \leq 2^{n-k}$.

Putting it Together. On input $(N, y = [x^e]_N)$, the RSA inversion algorithm A runs as follows. It applies Lemmas 2 and 3 to convert the (T, δ) distinguisher D into a $(T + O(m \log(e)n^2), \delta/(2m))$ distinguisher D' between distributions $\mathcal{D}'_{P,r,N}$ and $\mathcal{D}'_{R,r,N}$ which works for at least a fraction $\delta/2$ of $N \in \mathcal{I}_n$. Then A applies Lemma 4 to convert D' into the inversion algorithm A' . A now chooses random a and b in \mathbb{Z}_N . Since A does not know the ‘extra information’ $s_1 = L_r([ax]_N), u_1 = \widehat{M}_{N,k}([ax]_N), s_2 = L_r([bx]_N)$ and $u_2 = \widehat{M}_{N,l}([bx]_N)$ required by A' , A just exhaustively searches through all N_G possible values of (s_1, u_1, s_2, u_2) and runs A' on input $(N, y = [x^e]_N, \widehat{s}_1, \widehat{u}_1, \widehat{s}_2, \widehat{u}_2)$ for every guessed possibility $(\widehat{s}_1, \widehat{u}_1, \widehat{s}_2, \widehat{u}_2)$ until A' succeeds to recover x . Note that to find an approximation $\widehat{M}_{N,k}([ax]_N)$ correct within additive error $N/2^k$ it is enough to search through 2^{k-1} uniformly spaced possibilities $(N/2^{k-1})i$ for $i = 0, \dots, 2^{k-1} - 1$. Since $k = 3 \log(2mr/\delta) + 4 = 3 \log(2\ell/\delta) + 4$ and $l = \log(2\ell/\delta) + 4$, there are at most

$$N_G = 64(2\ell/\delta)^4 2^{2r} \tag{2}$$

guessing possibilities for $L_r([ax]_N), \widehat{M}_{N,k}([ax]_N), L_r([bx]_N), \widehat{M}_{N,l}([bx]_N)$ to search through. So the run-time bound of A is

$$\begin{aligned} T_{INV} &= N_G \cdot (4n \log(n)(2\ell/\delta)^2) \cdot (T + O(m \log(e)n^2)) \\ &= 2^{2r+14}(2\ell/\delta)^6 n \log(n) \cdot (T + O(m \log(e)n^2)). \end{aligned} \tag{3}$$

For at least a fraction $\delta/2$ of $N \in \mathcal{I}_n$, with the correct guessed value of the ‘extra information’, A' succeeds with probability at least $2/9$ over the choice of a, b . Hence we conclude that the success probability of A is at least $\epsilon_{INV} \geq \delta/9$, as claimed. \square

We can interpret Theorem 1 as follows. Suppose we assume that the expected run-time T_{INV}/ϵ_{INV} of any $(T_{INV}, \epsilon_{INV})$ RSA inversion algorithm is at least T_L . Then Theorem 1 can be used to convert a (T, δ) distinguisher for (n, e, r, ℓ) -RSAPRG to an RSA inverter contradicting our hardness assumption only if we output at most r bits per iteration, where

$$r < \frac{1}{2} \log \left(\frac{1}{9 \cdot 2^{14} \cdot n \log n \ell^6 \delta^{-7}} \cdot \frac{T_L}{T} \right). \tag{4}$$

Hence asymptotically, if we take $T_L = \text{poly}(n)$ (i.e. assume no poly-time RSA algorithm) then we get $r = O(\log(n))$ bits per iteration. If we assume that $T_L = O(2^{cn^{1/3}(\log n)^{2/3}})$ for constant c (run-time of the Number Field Sieve factoring algorithm [23]) then we can have $r = O(n^{1/3} \log^{2/3} n)$. But in any case, $r = o(n)$.

4 Our Modified Security Proof from an SSRSA Problem

We now explain how we modify the above reduction to solve a well-studied SSRSA problem and the resulting improved PRG efficiency/security tradeoff.

Our goal is to remove the search factor $N_G = 64 \cdot 2^{2r}(\ell/\delta)^4$ from the runtime bound (3) of the reduction in the proof of Theorem 1. The simplest way to do so is to provide the inversion algorithm A with the correct values for the ‘extra information’ required by the inversion algorithm A' of Lemma 4. This leads us to consider the following (not well-known) inversion problem that we call (n, e, r, k, l) -FSRSA :

Definition 4 ((n, e, r, k, l)-FSRSA Problem.). *Given RSA modulus N , and $(y = [x^e]_N, a \in_R \mathbb{Z}_N, s_1 = L_r([ax]_N), u_1 = \widehat{M}_k([ax]_N), b \in_R \mathbb{Z}_N, s_2 = L_r([bx]_N), u_2 = \widehat{M}_l([bx]_N))$, for $x \in_R \mathbb{Z}_N$, find x (here $\widehat{M}_{N,k}(x)$ denotes any approximation to x with additive error $|\widehat{M}_{N,k}(x) - x| \leq N/2^k$). We say that algorithm A is a (T, η) inversion algorithm for (n, e, r, k, l) -FSRSA if A runs in time at most T and has success probability at least η (over the random choice of $N \in_R \mathcal{I}_n, x, a, b \in_R \mathbb{Z}_N$ and the random coins of A , where \mathcal{I}_n is the same as in Definition 2).*

With the search factor N_G removed from the Fischlin-Schnorr reduction we therefore have that the hardness of the inversion problem (n, e, r, k, l) -FSRSA (with $k = 3 \log(2\ell/\delta) + 4$ and $l = \log(2\ell/\delta) + 4$) suffices for the ‘simultaneous security’ of the r least-significant RSA message bits (i.e. indistinguishability of distributions $\mathcal{D}'_{P,r,N}$ and $\mathcal{D}'_{R,r,N}$ in Lemma 3) and hence the pseudorandomness of (n, e, r, ℓ) -RSAPRG, with a much tighter reduction than the one of Theorem 1 relative to the RSA problem.

Theorem 2. *For all $n \geq 2^9$, any (T, δ) distinguisher D for (n, e, r, ℓ) -RSAPRG can be converted into a $(T_{INV}, \delta/9)$ inversion algorithm A for the (n, e, r, k, l) -FSRSA problem (with $k = 3 \log(2\ell/\delta) + 4$ and $l = \log(2\ell/\delta) + 4$) with run-time at most*

$$T_{INV} = 16 \cdot (\ell/\delta)^2 n \log(n) \cdot (T + O(\ell/r \log(e)n^2)). \tag{5}$$

Proof. We use the same inversion algorithm A as in the proof of Theorem 1, except that when applying Lemma 4, A runs inversion algorithm A' just once using the correct values of $(a, b, s_1 = L_r([ax]_N), u_1 = \widehat{M}_{N,k}([ax]_N), s_2 = L_r([bx]_N), u_2 = \widehat{M}_{N,l}([bx]_N))$ given as input to A , eliminating the search through $N_G = 64(2\ell/\delta)^4 2^{2r}$ possible values for (s_1, u_1, s_2, u_2) . \square

We defer to Section 6.1 our cryptanalysis of the (n, e, r, k, l) -FSRSA problem using the lattice-based method introduced by Coppersmith [11], which leads us to conjecture that the problem is hard whenever $r/n \leq 1/2 - 1/(2e) - (k + l)/2n - \epsilon$ for constant $\epsilon > 0$. This assumption together with the above reduction already implies the security of the efficient variants of (n, e, r, ℓ) -RSAPRG with $r = \Omega(n)$. Unfortunately, (n, e, r, k, l) -FSRSA is a new problem and consequently our conjecture on its hardness is not currently supported by extensive research. However, we will now show that in fact for $r/n = 1/2 - \max(k, l)/n - 1/e - \epsilon$ (note that this is smaller by $(\max(k, l) - (k + l)/2)/n + 1/(2e)$ than the largest secure value of r/n conjectured above), the problem (n, e, r, k, l) -FSRSA is at least as hard as a specific $(1/e + \epsilon, e)$ -SSRSA problem (i.e. with a specific univariate polynomial f of degree e) which we call (n, e, r, w) -CopRSA and define as follows:

Definition 5 ((n, e, r, w)-CopRSA Problem.). *Given RSA modulus N , and $(y = [x^e]_N, s_L = L_r(x), s_H = M_{n/2+w}(x))$, for $x \in_R \mathbb{Z}_N$, find x (here $M_k(x)$ denotes the k most-significant bits of the binary representation of x). We say that algorithm A is a (T, η) inversion algorithm for (n, e, r, w) -CopRSA if A runs in time at most T and has success probability at least η (over the random choice of $N \in_R \mathcal{I}_n$, $x \in_R \mathbb{Z}_N$ and the random coins of A , where \mathcal{I}_n is the same as in Definition 2).*

To see that (n, e, r, w) -CopRSA problem is a specific type of SSRSA problem, note that it is equivalent to finding a small solution $\bar{z} < 2^{n/2-(r+w)}$ (consisting of bits $r + 1, \dots, (n/2 - w)$ of the randomly chosen integer x) to the equation $f(\bar{z}) \equiv y \pmod N$, where the degree e polynomial $f(z) = (2^r z + s)^e$, where $s = s_H \cdot 2^{n/2-w} + s_L$ is known. Hence (n, e, r, w) -CopRSA is a $(1/e + \epsilon, e)$ -SSRSA problem when $1/2 - (r + w)/n = 1/e + \epsilon$, i.e. $r/n = 1/2 - 1/e - \epsilon - w/n$.

Theorem 3. *Let A' be a (T', η') attacker against $(n, e, r, w - 1, w - 1)$ -FSRSA. Then we construct a (T, η) attacker A against (n, e, r, w) -CopRSA with*

$$T = 4T' + O(n^2) \text{ and } \eta = \eta' - 4/2^{n/2}.$$

Proof. On input $(N, y = [x^e]_N, s_L = L_r(x), s_H = M_{n/2+w}(x))$, for $N \in_R \mathcal{I}_n$ and $x \in_R \mathbb{Z}_N$, the attacker A runs as follows:

- Choose a uniformly random $b \in_R \mathbb{Z}_N$.
- Compute an integer c coprime to N with $|c| < N^{1/2}$ such that $|[b \cdot c]_N| < N^{1/2}$ (here $[z]_N \in (-N/2, N/2)$ denotes the ‘symmetrical’ residue of z modulo N , i.e. $[z]_N \stackrel{\text{def}}{=} [z]_N$ if $[z]_N \in [0, N/2)$ and $[z]_N \stackrel{\text{def}}{=} [z]_N - N$ if $[z]_N \in (N/2, N)$). It is well known that such a c exists and can be computed efficiently (in time $O(n^2)$) using continued fractions (see, e.g. Lemma 16 in [25]).
- Observe that $[cx]_N = cx - \omega_c N$, where $\omega_c = \lfloor \frac{cx}{N} \rfloor$. Let $\hat{x} = s_H \cdot 2^{n/2-w}$. Notice that \hat{x} approximates x within additive error $\Delta_x \leq 2^{n/2-w}$ and consequently the rational number $\frac{c\hat{x}}{N}$ approximates $\frac{cx}{N}$ within additive error $\frac{|c|\Delta_x}{N} \leq \Delta_x/N^{1/2} \leq 2^{n/2-w}/2^{(n-1)/2} < 1$, where we have used the fact that $|c| < N^{1/2}$ and $w \geq 1$. It follows that $\omega_c \in \{\lfloor \frac{c\hat{x}}{N} \rfloor, \lfloor \frac{c\hat{x}}{N} \rfloor \pm 1\}$ (where the $+$ sign applies if $c \geq 0$ and the $-$ sign applies otherwise). So **A** obtains 2 candidates for ω_c .
- Using $L_r([cx]_N) = L_r(cx - \omega_c N) = L_r(L_r(c) \cdot L_r(x) - L_r(\omega_c N))$, **A** computes (with the known $s_L = L_r(x)$, c and N) 2 candidates for $L_r([cx]_N)$ from the 2 candidates for ω_c .
- Similarly, writing $[bcx]_N = [bc]_N \cdot x - \omega_{bc} N$, with $\omega_{bc} = \lfloor \frac{[bc]_N x}{N} \rfloor$, using $|[bc]_N| < N^{1/2}$ we obtain $\omega_{bc} \in \{\lfloor \frac{[bc]_N \hat{x}}{N} \rfloor, \lfloor \frac{[bc]_N \hat{x}}{N} \rfloor \pm 1\}$ (with $+$ sign if $[bc]_N \geq 0$ and $-$ sign otherwise), so **A** also computes 2 candidates for ω_{bc} and two corresponding candidates for $L_r([bcx]_N) = L_r([bc]_N x - \omega_{bc} N) = L_r(L_r([bc]_N) L_r(x) - \omega_{bc} N)$.
- Using \hat{x} and the 2 candidates for ω_c computed above, **A** computes two candidate approximations $c\hat{x} - \omega_c N$ for $[cx]_N$. Since \hat{x} approximates x within additive error $\Delta_x \leq 2^{n/2-w}$ we have that $c\hat{x} - \omega_c N$ approximates $[cx]_N$ within additive error $|c|\Delta_x \leq N^{1/2} 2^{(n-1)/2} / 2^{w-1/2} \leq N/2^{w-1}$ using $N \geq 2^{n-1}$.
- Similarly, using \hat{x} and the 2 candidates for ω_{bc} computed above, **A** computes two candidate approximations $[bc]_N \hat{x} - \omega_{bc} N$ for $[bcx]_N$, one of which has additive error $|[bc]_N| \Delta_x \leq N/2^{w-1}$.
- Choose a uniformly random $a \in \mathbb{Z}_N^*$ and compute $y' = [(a^{-1}c)^e y]_N = [(a^{-1}cx)^e]_N$.
- Collecting all of the above information, **A** obtains 4 candidates for $(N, y' = [(a^{-1}cx)^e]_N, a, s_1 = L_r([cx]_N), u_1 = \widehat{M}_{N, w-1}([cx]_N), b' = [ab]_N, s_2 = L_r([bcx]_N), u_2 = \widehat{M}_{N, w-1}([bcx]_N))$. Note that this is a valid instance of $(n, e, r, w-1, w-1)$ -FSRSA. Furthermore, it has almost exactly the correct distribution, since the triple $(x' = [a^{-1}cx]_N, a, b' = [ab]_N)$ is uniformly random in $\mathbb{Z}_N \times \mathbb{Z}_N^* \times \mathbb{Z}_N$ thanks to the uniformly random choice of $(x, a, b) \in \mathbb{Z}_N \times \mathbb{Z}_N^* \times \mathbb{Z}_N$. The FSRSA instance distribution is not exactly correct because here a is uniform on \mathbb{Z}_N^* while it should be uniform on \mathbb{Z}_N . However, simple calculation shows that the statistical distance between the uniform distribution on \mathbb{Z}_N^* and the uniform distribution on \mathbb{Z}_N is negligible, namely $1 - \phi(N)/N = (p+q-1)/N \leq 4/2^{n/2}$.
- **A** runs **A'** on the above 4 candidate $(n, e, r, w-1, w-1)$ -FSRSA instances. On one of those runs, **A'** outputs $x' = [a^{-1}cx]_N$ with probability at least $\eta - 4/2^{n/2}$, from which x is easily recovered as $x = [ac^{-1}x']_N$.

Note that the run-time of A is bounded as $T \leq 4T' + O(n^2)$ and A succeeds with probability at least $\eta - 4/2^{n/2}$, as required. This completes the proof. \square

So, combining Theorems 2 and 3, we conclude:

Corollary 1. *For all $n \geq 2^9$, any (T, δ) distinguisher D for (n, e, r, ℓ) -RSAPRG can be converted into a $(T_{INV}, \epsilon_{INV})$ inversion algorithm A for the (n, e, r, w) -CopRSA problem (with $w = 3 \log(2\ell/\delta) + 5$) with*

$$T_{INV} = 64 \cdot (\ell/\delta)^2 n \log(n) \cdot (T + O(\ell/r \log(e)n^2)) \text{ and } \epsilon_{INV} = \delta/9 - 4/2^{n/2}. \quad (6)$$

Remark. Fischlin and Schnorr [14] also outline an alternative security reduction (worked out in detail and optimized for the Rabin iteration function by Sidorenko and Schoenmakers [28]) for the (n, e, r, ℓ) -RSAPRG with $r > 1$ based on a general ‘Computational XOR Lemma’ [30,16]. However, this alternative reduction has an inherent exponential run-time factor 2^{2r} which we do not know how to eliminate, even using our stronger SSRSA assumption on RSA inversion.

5 Concrete Parameters and Estimated Performance

Using (6) we obtain an upper bound on the pseudorandom string length ℓ for a given security level (T, δ) and assumed expected run-time lower bound T_L for breaking the $(n, e, r, 3 \log(2\ell/\delta) + 5)$ -CopRSA problem. Recall that the latter is a $(1/e + \epsilon, e)$ -SSRSA problem when

$$r/n = 1/2 - 1/e - \epsilon - (3 \log(2\ell/\delta) + 5)/n, \quad (7)$$

and that $(1/e + \epsilon, e)$ -SSRSA problem is conjectured to take time $T_L = \min(T_F(n), T_C(n, \epsilon))$, where $T_F(n)$ is a lower bound for factoring N and $T_C(n, \epsilon) = \text{poly}(n) \cdot 2^{\epsilon n}$ is the time for the Coppersmith attack on $(1/e + \epsilon, e)$ -SSRSA. Asymptotically, we therefore have for any constant $\epsilon > 0$ that $T_L = T_F(n)$ since $T_F(n)$ is subexponential in n , so for any $\ell/\delta = \text{poly}(n)$ and $e \geq 3$ we can use $r/n = 1/2 - 1/e - \epsilon - o(1)$, i.e. $r = \Omega(n)$. The exact bound on r for a given modulus length n depends on the value of ϵ such that $T_F(n) = T_C(n, \epsilon)$. To estimate concrete values, we used the Number Field Sieve (NFS) factoring run-time model from [23] (we refer to the full version of the paper for more details [29]) – the results are summarised in Table 1.

Our estimates indicate that we can (with $n = 6144$ bit and $e = 8$) achieve a rate around 19300 cycles/byte (0.87 Mbit/s with 2.1 GHz clock) on a Pentium 4 Processor, outputting more than 2^{30} bits with provable 2^{70} instructions distinguishing run-time (under the $(1/e + \epsilon, e)$ -SSRSA assumption). This seems to be close to practical requirements of some stream cipher applications (it is several hundred times faster than the basic Blum-Blum Shub generator outputting one bit per iteration with the same modulus length). Compared to the recent provably secure QUAD PRG construction [3] (based on the ‘MQ’ problem), our PRG seems to have a lower throughput, although it is difficult to make a fair comparison since unlike our figures above, the performance figures reported in [3]

Table 1. Estimate of achievable performance for provable $T = 2^{70}$ instructions distinguishing time to achieve advantage $\delta = \frac{1}{100}$, using $e = 8, 9$ (assuming hardness of the CopRSA SSRSA problem) and $e = 2$ (assuming hardness of FSRSA problem - see Section 6). Throughput ('Thrpt') columns are estimated throughput based on Wei Dai's Crypto++ benchmarks page [13] (for Pentium 4 2.1GHz processor) and extrapolation assuming classical arithmetic.

n (bit)	$\log(\ell)$	Rate, $e = 8$ (bit/mult)	Thrpt (Mbit/s)	Rate, $e = 9$ (bit/mult)	Thrpt (Mbit/s)	Rate, $e = 2$ (bit/mult)	Thrpt (Mbit/s)
3072	9.3	341	1.68	267	1.31	660	3.2
4096	18.0	460	1.28	360	1.00	899	2.5
5120	25.4	581	1.03	454	0.80	1140	2.0
6144	32.0	702	0.87	549	0.67	1383	1.7

(between 3000 and 4500 cycles/byte on Pentium 4) are for a 'practical' choice of parameters, smaller than those for which the security proof can be applied. A possible advantage of our construction is its significantly smaller static parameters (i.e. non-secret parameters defining the pseudorandom generator) of length $n \approx 5$ kbit, while in [3] the static parameters are longer than 1 Mbit (this might allow our construction to be implemented with less code memory requirements). On the other hand, our construction has a longer state and is based on the hardness of factoring so is insecure against potential future quantum attacks, while the MQ problem in [3] may be secure even against such attacks.

6 Potential Improvements

6.1 Cryptanalysis of the FS-RSA Problem

As observed in Section 4, the (n, e, r, k, l) -FSRSA problem, although not well-known, gives a more direct proof of security for the RSA PRG than the SSRSA problem. In this section we describe a 'Coppersmith-type' lattice attack on (n, e, r, k, l) -FSRSA (which we believe is essentially optimal) and show that it is likely to succeed only when $r/n \geq 1/2 - (k+l)/(2n) - 1/(2e)$. This value of r/n is larger by about $1/(2e) + (\max(k, l)/n - (k+l)/(2n))$ than that the largest value for which the corresponding SSRSA problem in Section 4 is secure, leading to improved throughput for the RSA PRG by using this stronger assumption.

The attack on (n, e, r, k, l) -FSRSA problem works as follows. First we reduce the problem to solving two modular equations in two small unknowns z_1 and z_2 . Namely, given $(y = [x^e]_N, a \in_R \mathbf{Z}_N, s_1 = L_r([ax]_N), u_1 = \widehat{M}_{N,k}([ax]_N), b \in_R \mathbf{Z}_N, s_2 = L_r([bx]_N), u_2 = \widehat{M}_{N,l}([bx]_N))$, we have

$$x^e \equiv y \pmod{N}, \tag{8}$$

$$[ax]_N = s_1 + \bar{z}'_1 \cdot 2^r; |[ax]_N - u_1| \leq N/2^k \tag{9}$$

and

$$[bx]_N = s_2 + \bar{z}'_2 \cdot 2^l; |[bx]_N - u_2| \leq N/2^l \tag{10}$$

where $\bar{z}'_1 < N/2^r$ and $\bar{z}'_2 < N/2^r$ consist of the $n - r$ MS bits of $[ax]_N$ and $[bx]_N$, respectively. Let $\hat{z}_1 = \lfloor \frac{u_1 - s_1}{2^r} \rfloor$. From (9) we conclude that $|\bar{z}'_1 - \hat{z}_1| \leq |(\frac{[ax]_N - s_1}{2^r}) - (\frac{u_1 - s_1}{2^r})| + 1 \leq N/2^{r+k} + 1 \leq N/2^{r+k-1}$ (for $2^{r+k} < N$) and hence letting $\bar{z}_1 = \bar{z}'_1 - \hat{z}_1$ we obtain $[ax]_N = (s_1 + 2^r \hat{z}_1) + 2^r \bar{z}_1$ where integer $|\bar{z}_1| < N/2^{r+k-1}$. Similarly, from (10) we obtain $[bx]_N = (s_2 + 2^r \hat{z}_2) + 2^r \bar{z}_2$ where integer $|\bar{z}_2| < N/2^{r+l-1}$ (for $2^{r+l} \leq N$) and $\hat{z}_2 = \lfloor (u_2 - s_2)/2^r \rfloor$. Treating the last two equations for $[ax]_N$ and $[bx]_N$ as congruences modulo N , we eliminate the unknown variable x (by multiplying the second congruence by $[ab^{-1}]_N$ and subtracting from the first) to obtain a single linear polynomial $f(z_1, z_2)$ in two variables z_1, z_2 , having the desired small unknowns \bar{z}_1, \bar{z}_2 as a zero modulo N (i.e. $f(\bar{z}_1, \bar{z}_2) \equiv 0 \pmod{N}$), namely:

$$f(z_1, z_2) = \alpha \cdot z_1 + z_2 + \beta, \tag{11}$$

where $\alpha = [-ab^{-1}]_N$ and $\beta = [-a^{-1}b2^{-r}(s_1 + 2^r \hat{z}_1) + 2^{-r}(s_2 + 2^r \hat{z}_2)]_N$ are known. Also, substituting $x \equiv a^{-1}(s_1 + 2^r \hat{z}_1) + 2^r a^{-1} \hat{z}_1 \pmod{N}$ into (8) we obtain a degree e univariate polynomial in z_1 having the small unknown \bar{z}_1 as a zero modulo N (i.e. $g(\bar{z}_1) \equiv 0 \pmod{N}$):

$$g(z_1) = (z_1 + \hat{\alpha})^e - \hat{\beta}, \tag{12}$$

where $\hat{\alpha} = [2^{-r}s_1 + \hat{z}_1]_N$ and $\hat{\beta} = [-(a2^{-r})^e y]_N$ are known. To find the small zero (\bar{z}_1, \bar{z}_2) of (11) and (12) we use the bivariate modular polynomial lattice method of Coppersmith [11] as simplified by Howgrave-Graham [20] and used in many subsequent works. Namely, for an integer m we use the polynomials $f(z_1, z_2)$ and $g(z_1)$ to construct the following family of polynomials $h_{i,k}(z_1, z_2)$ indexed by a pair of integers $i = 0, 1, \dots, me$ (which we refer to as the ‘block index’) and $k = 0, \dots, i$ (which we call the ‘inner index’) for each block i :

$$h_{i,k}(z_1, z_2) = N^{me - (i-k + \lfloor \frac{k}{e} \rfloor)} z_1^{\lfloor \frac{k}{e} \rfloor} g(z_1)^{\lfloor \frac{k}{e} \rfloor} f(z_1, z_2)^{i-k}. \tag{13}$$

Observe that each of the polynomials $h_{i,k}(z_1, z_2)$ has (\bar{z}_1, \bar{z}_2) as a zero modulo N^{me} , because $f(\bar{z}_1, \bar{z}_2)^{i-k} \equiv 0 \pmod{N^{i-k}}$ and $g(\bar{z}_1)^{\lfloor \frac{k}{e} \rfloor} \equiv 0 \pmod{N^{\lfloor \frac{k}{e} \rfloor}}$.

It follows that any integer linear combination of the polynomials $h_{i,k}(z_1, z_2)$ also has (\bar{z}_1, \bar{z}_2) as a zero modulo N^{me} . Let $B_1 = N/2^{r+k-1}$ and $B_2 = N/2^{r+l-1}$ denote the upper bounds derived above on $|\bar{z}_1|$ and $|\bar{z}_2|$, respectively. We set up a lattice \mathcal{L} to search for linear combinations of the polynomials $h_{i,k}(z_1, z_2)$, which have sufficiently small coefficients such that they have (\bar{z}_1, \bar{z}_2) as a zero over the integers, not just modulo N^{me} . Given two such linearly independent polynomials we can take their resultant to obtain a single univariate polynomial equation in z_1 over the integers which is easy to solve. The square basis matrix $\mathcal{B}_{\mathcal{L}}$ for lattice \mathcal{L} has rows and columns indexed by pairs of integers (i, k) , where the (i', k') th column of the (i, k) th row of $\mathcal{B}_{\mathcal{L}}$ contains the coefficient of the monomial $z_1^{k'} z_2^{i'-k'}$ in the polynomial $h_{i,k}(B_1 z_1, B_2 z_2)$. With this ordering, $\mathcal{B}_{\mathcal{L}}$ is in lower diagonal form and its determinant $\det(\mathcal{L})$ is the product of the diagonal elements of $\mathcal{B}_{\mathcal{L}}$. Some straightforward calculations (see full paper [29]) show that $\det(\mathcal{L}) = N^{me \cdot d(me) - W(m,e)} (B_1 B_2)^{D(me)/2}$, where $d(me) = \frac{1}{2}(me + 1)(me + 2)$ is

the dimension of \mathcal{L} , $D(me) = \frac{e^3}{3}m^3 + O(m^2)$ and $W(m, e) = \frac{1}{2}D(me) + \frac{e^2}{6}m^3 + O(m^2)$. Let $h_1(z_1, z_2)$ and $h_2(z_1, z_2)$ denote the polynomials corresponding to the first two vectors in the reduced basis of \mathcal{L} returned by LLL on input $\mathcal{B}_{\mathcal{L}}$. Using Lemma 1, we can show (see full paper [29]) that h_1 and h_2 will have a common zero over \mathbb{Z} if the following condition is satisfied:

$$2^{d(me)/2} \det(\mathcal{L})^{\frac{1}{d(me)-1}} < \frac{N^{me}}{\sqrt{d(me)}}. \tag{14}$$

Plugging the expression for $\det(\mathcal{L})$ into this condition, we obtain $(B_1 B_2)^{1/2} < N^{\frac{W(m,e)-me}{D(me)}} / \gamma(me)$, where the factor $\gamma(me) \stackrel{\text{def}}{=} (\sqrt{d(me)} 2^{d(me)/2})^{\frac{d(me)-1}{D(me)}}$ is independent of n and so is of order $O(N^{o(1)})$ as n increases. For increasing parameter m , the leading m^3 terms dominate, and hence the ratio $\frac{W(m,e)-me}{D(me)}$ approaches asymptotically the value $\frac{1}{2} + \frac{e^2/6}{e^3/3} = \frac{1}{2} + \frac{1}{2e}$. So the attack success condition becomes $(B_1 B_2)^{1/2} < N^{1/2+1/(2e)-o(1)}$ for large n and m . Using $B_1 = \frac{N}{2^{r+k-1}}$ and $B_2 = \frac{N}{2^{r+l-1}}$ and $N < 2^n$ we obtain the asymptotic attack success bound

$$\frac{r}{n} > 1/2 - 1/(2e) - \frac{(k+l)}{2n} + o(1). \tag{15}$$

Although the attack is heuristic (in the sense that resultant of h_1 and h_2 may be a zero polynomial), our numerical experiments (see [29]) suggest that the attack works in practice. We conjecture that bound (15) is essentially optimal for ‘Coppersmith-type’ lattice attacks on (n, e, r, k, l) -FSRSA (see [29]).

6.2 Using Even Exponents

Assuming Hardness of FSRSA Problem. If we assume that the attack of the previous section is optimal so the (n, e, r, k, l) -FSRSA problem is hard when the bound (15) is violated, then we can allow r/n to approach $1/4$ even for $e = 2$, with only one modular squaring required per iteration. It is shown in [14] that with appropriate modifications to the proof, Lemma 4 holds also for $e = 2$ if we replace the iteration function $x \rightarrow [x^e]_N$ by the ‘absolute Rabin function’ $f_a(x) = [x^2]_N \stackrel{\text{def}}{=} \min([x^2]_N, N - [x^2]_N)$, choose $N = pq$ to be a *Blum* RSA modulus with $p \equiv q \equiv 3 \pmod{4}$, and choose the PRG seed $x_0 \in_R M_N$, where $M_N \stackrel{\text{def}}{=} \mathbb{Z}_N^*(+1) \cap (0, N/2)$, and $\mathbb{Z}_N^*(+1)$ denotes the subset of elements of \mathbb{Z}_N^* having Jacobi symbol $+1$. Since f_a permutes the set M_N , the proof of Lemma 3 holds as well. Refer to Table 1 for performance of this PRG variant, where it is assumed that the best attack on (n, e, r, k, l) -FSRSA with $r/n = 1/2 - 1/(2e) - \frac{(k+l)}{2n} + \epsilon$ takes time $\min(T_F(n), 2^{\epsilon n})$, where $T_F(n)$ is the time needed to factor N . We stress however that this assumption is new and needs further study.

Assuming Hardness of SSRSA Problem. Our reduction (Theorem 3) from the CopRSA to FSRSA problem also extends with some small modifications to the case of even e (see [29]). For $e = 8$, it actually gives better rate than the best odd exponent assuming the hardness of SSRSA ($e = 9$) – see Table 1.

7 Conclusion

We have shown that an efficient variant of the RSA PRG is provably secure assuming the hardness of a well-studied variant of the RSA inversion problem in which some of the plaintext bits are known.

We see two avenues for further improvement. Even using the FSRSA assumption in Section 6, the PRG rate which we can prove secure is $r = (1/2 - 1/(2e) - \epsilon - o(1))n$ for ‘small’ ϵ . Can this rate be improved using a different proof (but a similar inversion assumption) up to $r = (1 - 1/e - \epsilon - o(1))n$? The other question is whether the factor ℓ^2 in the reduction run-time factor $O((\ell/\delta)^2 n \log(n))$ can be significantly reduced.

Finally we remark that besides generic applications of PRGs, our result can also be applied to prove security of an efficient semantically secure (IND-CPA) RSA-based public key encryption scheme, assuming the hardness of the SSRSA one-wayness problem (see [29]). An interesting open problem is to construct additional efficient cryptographic primitives based on this problem.

Acknowledgements. The authors would like to thank Scott Contini and Igor Shparlinski for enlightening discussions and encouragement and the anonymous referees for their useful comments. This work was supported by Australian Research Council Discovery Grants DP0663452, DP0451484 and DP0665035.

References

1. W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA and Rabin Functions: Certain Parts Are as Hard as the Whole. *SIAM Journal on Computing*, 17(2):194–209, 1988.
2. M. Ben-Or, B. Chor, and A. Shamir. On the Cryptographic Security of Single RSA Bits. In *Proc. 15-th STOC*, pages 421–430, New York, 1983. ACM Press.
3. C. Berbain, H. Gilbert, and J. Patarin. QUAD: a Practical Stream Cipher with Provable Security. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 109–128, Berlin, 2006. Springer-Verlag.
4. S.R. Blackburn, D. Gomez-Perez, J. Gutierrez, and I.E. Shparlinski. Reconstructing Noisy Polynomial Evaluation in Residue Rings. *Journal of Algorithms*. (To Appear).
5. S.R. Blackburn, D. Gomez-Perez, J. Gutierrez, and I.E. Shparlinski. Predicting Nonlinear Pseudorandom Number Generators. *Mathematics of Computation*, 74:1471–1494, 2004.
6. L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, 15:364–383, 1986.
7. M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, 13:850–864, 1984.
8. D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Trans. on Info. Theory*, 46(4):1339–1349, 2000.
9. D. Boneh, S. Halevi, and N.A. Howgrave-Graham. The Modular Inversion Hidden Number Problem. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 36–51, Berlin, 2001. Springer-Verlag.

10. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's Cryptosystem Revisited. In *Proc. CCS '01*, New York, November 2001. ACM.
11. D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. of Cryptology*, 10:233–260, 1997.
12. D. Coppersmith. Finding Small Solutions to Low Degree Polynomials. In *CALC '01*, volume 2146 of *LNCS*, pages 20–31, Berlin, 2001. Springer-Verlag.
13. W. Dai. *Crypto++ 5.2.1 Benchmarks*, 2006. <http://www.eskimo.com/~weidai/benchmarks.html>.
14. R. Fischlin and C.P. Schnorr. Stronger Security Proofs for RSA and Rabin Bits. *Journal of Cryptology*, 13:221–244, 2000.
15. R. Gennaro. An Improved Pseudo-Random Generator Based on the Discrete-Logarithm Problem. *Journal of Cryptology*, 18:91–110, 2005.
16. O. Goldreich. *Foundations of Cryptography, Volume I*. Cambridge University Press, Cambridge, 2003.
17. O. Goldreich and V. Rosen. On the Security of Modular Exponentiation with Application to the Construction of Pseudorandom Generators. *J. of Cryptology*, 16:71–93, 2003.
18. S. Goldwasser and S. Micali. Probabilistic Encryption. *J. of Computer and System Sciences*, 28(2):270–299, 1984.
19. S. Goldwasser, S. Micali, and P. Tong. Why and How to Establish a Private Code on a Public Network. In *Proc. FOCS '82*, pages 134–144. IEEE Computer Society Press, 1982.
20. N. Howgrave-Graham. Finding Small Roots of Univariate Polynomials Revisited. In *Cryptography and Coding*, volume 1355 of *LNCS*, pages 131–142, Berlin, 1997. Springer-Verlag.
21. R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *Journal of Cryptology*, 9:199–216, 1996.
22. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261:515–534, 1982.
23. A.K. Lenstra and E.R. Verheul. Selecting Cryptographic Key Sizes. *J. of Cryptology*, 14:255–293, 2001.
24. S. Micali and C.P. Schnorr. Efficient, Perfect Polynomial Random Number Generators. *J. of Cryptology*, 3:157–172, 1991.
25. P. Q. Nguyen and I. E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15:151–176, 2002.
26. P. Q. Nguyen and J. Stern. The Two Faces of Lattices in Cryptology. In *Cryptography and Lattices*, volume 2146 of *LNCS*, pages 146–180, Berlin, 2001. Springer-Verlag.
27. S. Patel and G. Sundaram. An Efficient Discrete Log Pseudo Random Generator. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 304–317, Berlin, 1998. Springer-Verlag.
28. A. Sidorenko and B. Schoenmakers. Concrete Security of the Blum-Blum-Shub Pseudorandom Generator. In *Cryptography and Coding 2005*, volume 3796 of *LNCS*, pages 355–375, Berlin, 2005. Springer-Verlag.
29. R. Steinfeld, J. Pieprzyk, and H. Wang. On the Provable Security of an Efficient RSA-Based Pseudorandom Generator. Cryptology ePrint Archive, Report 2006/206, 2006. <http://eprint.iacr.org/2006/206>.
30. U.V. Vazirani and V.V. Vazirani. Efficient and Secure Pseudo-Random Number Generation. In *Proc. FOCS '84*, pages 458–463. IEEE Computer Society Press, 1982.

On the Security of OAEP

Alexandra Boldyreva¹ and Marc Fischlin²

¹ College of Computing, Georgia Institute of Technology, USA

sasha@gatech.edu

www.cc.gatech.edu/~aboldyre

² Darmstadt University of Technology, Germany

marc.fischlin@gmail.com

www.fischlin.de

Abstract. Currently, the best and only evidence of the security of the OAEP encryption scheme is a proof in the contentious random oracle model. Here we give further arguments in support of the security of OAEP. We first show that partial instantiations, where one of the two random oracles used in OAEP is instantiated by a function family, can be provably secure (still in the random oracle model). For various security statements about OAEP we specify sufficient conditions for the instantiating function families that, in some cases, are realizable through standard cryptographic primitives and, in other cases, may currently not be known to be achievable but appear moderate and plausible. Furthermore, we give the first non-trivial security result about *fully* instantiated OAEP *in the standard model*, where both oracles are instantiated simultaneously. Namely, we show that instantiating both random oracles in OAEP by modest functions implies non-malleability under chosen plaintext attacks for random messages. We also discuss the implications, especially of the full instantiation result, to the usage of OAEP for secure hybrid encryption (as required in SSL/TLS, for example).

1 Introduction

OAEP is one of the most known and widely deployed asymmetric encryption schemes. It was designed by Bellare and Rogaway [5] as a scheme based on a trapdoor permutation such as RSA. OAEP is standardized in RSA's PKCS #1 v2.1 and is part of the ANSI X9.44, IEEE P1363, ISO 18033-2 and SET standards. The encryption algorithm of $\text{OAEP}^{G,H}[F]$ takes a public key f , which is an instance of a trapdoor permutation family F , and a message M , picks r at random and computes the ciphertext $C = f(s||t)$ for $s = G(r) \oplus M||0^{k_1}$ and $t = H(s) \oplus r$, where G and H are some hash functions. Despite its importance the only security results for OAEP are a proof of IND-CPA security assuming F is a one-way trapdoor permutation family [5] and a proof of IND-CCA2 security assuming F is partial one-way [16], both in the random oracle (RO) model, i.e., where G and H are idealized and modeled as random oracles [4]. However, such proofs merely provide heuristic evidence that breaking the scheme may be hard in reality (when the random oracles are instantiated with real functions).

A growing number of papers raised concerns regarding soundness of the controversial random oracle model [12,19,20,17,1,14,9,21]. Moreover, most of the recent results question security of the practical schemes known to be secure in the RO model. For example, Dodis et al. [14] showed some evidence that the RSA Full Domain Hash signature scheme may not be secure in the standard model. Boldyreva and Fischlin [9] showed that even presumably strong candidates like perfectly one-way hash functions (POWHFs) [11,13] are insufficient to prove security of partial instantiations of OAEP (when only one of the two random oracles is instantiated with an instance of a POWHF).

The motivation of this work is to gather evidence of soundness of the OAEP design. Like the aforementioned works our goal is to go beyond the classical RO heuristic and study security of the scheme when one or all of its ROs are instantiated. Positive results in the direction of partial instantiations would give further evidence that breaking OAEP for good instantiations is hard, because breaking the scheme would then require to exploit interdependent weaknesses between the instantiations or the family F . Given the negative results of [9] it is unlikely to expect that the properties needed from the instantiating function families are weak or even easily realizable, even if one accepts weaker security stipulations than chosen-ciphertext security for partial or full instantiations. For example, although it seems plausible, it is currently not even known whether OAEP can be proven IND-CPA secure in the standard model assuming any reasonable properties of the instantiating functions.

Here we show that security proofs for instantiations of OAEP are indeed possible. For various security statements about OAEP we specify sufficient conditions on G and H that are certainly weaker than assuming that the functions behave as random oracles, yielding “positive” security statements regarding partially instantiated OAEP. Furthermore, we give the first non-trivial security results about fully instantiated OAEP in the standard model, where both oracles G and H are instantiated simultaneously. We next discuss these results in more detail.

THE OAEP FRAMEWORK. For better comprehension of our technical results we first reconsider the OAEP encryption scheme from a more abstract viewpoint. Let f be a random instance of a partial one-way trapdoor permutation family F , and the encryption algorithm computes a ciphertext as $C = f(s||t)$. Partial one-wayness [16] requires that it is hard to find the leading part of the pre-image $s||t$ under f and to output, say, s only. If we consider now for example a family $F_{\text{t-clear}}$ where each function is defined as $f \equiv g||\text{ID}$ such that $f(s||t) = g(s)||t$ for a trapdoor permutation g , then this family $F_{\text{t-clear}}$ is clearly partial one-way (and also a trapdoor permutation). Hence, this example describes a special case $\text{OAEP}^{G,H}[F_{\text{t-clear}}]$ for the partial one-way trapdoor permutation family $F_{\text{t-clear}}$ where each function outputs the t -part in clear. In particular, the security proof in the random oracle model for OAEP and general partial one-way families (including RSA as a special case) [16] carries over, but we outdo this by giving positive results of partial instantiation for such families $F_{\text{t-clear}}$.

Towards the standard-model security results for fully instantiated OAEP we take the above viewpoint one step further and look at $\text{OAEP}^{G,H}[F_{\text{lsb}||\text{t-clear}}]$ for

families $F_{\text{lsb}||t\text{-clear}}$ where each function f outputs the k_1 least significant bits of $s = G(r) \oplus M||0^{k_1}$ (which equal those bits of $G(r)$) and t in clear. Since each function in $F_{\text{lsb}||t\text{-clear}}$ is also a member in $F_{t\text{-clear}}$ the partial instantiation results above remain true for $\text{OAEP}^{G,H}[F_{\text{lsb}||t\text{-clear}}]$.

We note that security of partial instantiations of $\text{OAEP}^{G,H}[F_{t\text{-clear}}]$ and of $\text{OAEP}^{G,H}[F_{\text{lsb}||t\text{-clear}}]$, although for qualified partial one-way trapdoor families, also have implications for the popular $\text{OAEP}^{G,H}[\text{RSA}]$ case. They show that any successful attacks on instantiations for RSA would have to take advantage of specific properties of the RSA function. Generic attacks which would also work for $F_{t\text{-clear}}$ or $F_{\text{lsb}||t\text{-clear}}$ are then ruled out.

PARTIAL INSTANTIATION RESULTS. Positive results about partial instantiations were first shown in [9] for the PSS-E encryption scheme. There it was also shown, however, that perfectly one-way hash functions cannot be securely used to instantiate either one of the ROs in OAEP. These negative results about partial instantiation through POWHFs hold for $\text{OAEP}^{G,H}[F_{t\text{-clear}}]$ as well. Yet we show that partial instantiations are possible by switching to other primitives.

To instantiate the G -oracle in $\text{OAEP}^{G,H}[F_{t\text{-clear}}]$ while preserving IND-CCA2 security (in the random oracle model), we introduce the notion of a near-collision resistant pseudorandom generator. For such a generator G it is infeasible to find different seeds $r \neq r'$ such that predetermined parts of the generator's outputs $G(r)$, $G(r')$ match (they may differ on other parts). To be more precise for $\text{OAEP}^{G,H}[F_{t\text{-clear}}]$ the generator G is not allowed to coincide on the k_1 least significant bits, bequeathing this property to the values $s = G(r) \oplus M||0^{k_1}$ and $s' = G(r') \oplus M||0^{k_1}$ in the encryption process. We discuss that such pseudorandom generators can be derived from any one-way permutation.

Instantiating the H oracle in OAEP turns out to be more challenging. To this end we consider non-malleable pseudorandom generators, where a given image of a seed r should not help significantly to produce an image of a related seed r' . Instantiating H through such a non-malleable pseudorandom generator the resulting scheme achieves NM-CPA security, where it is infeasible to convert a given ciphertext into one of a related message. Although this security notion for encryption schemes is not as strong as IND-CCA, it yet exceeds the classical IND-CPA security. That is, Bellare et al. [3] show that NM-CPA implies IND-CPA and is incomparable to IND-CCA1 security. Hence, NM-CPA security of schemes lies somewhere in between IND-CPA and IND-CCA2.¹

We also show that it is possible to extend the above result and to instantiate the H -oracle in $\text{OAEP}^{G,H}[F_{t\text{-clear}}]$ without even sacrificing IND-CCA2 security (again, for random oracle G). This however requires the very strong assumption for the pseudorandom generators which then must be non-malleable under chosen-image attacks. For such a generator non-malleability should even hold if the adversary can learn seeds of chosen images, and such generators resemble

¹ We mitigate the notion of NM-CPA such that the relation specifying related messages and the distribution over the messages must be fixed at the outset. This mildly affects the relationship to the IND notions, but we omit technical details in the introduction.

chosen-ciphertext secure encryption schemes already. Hence, we see this partial instantiation as a mere plausibility result that one can presumably instantiate oracle H and still have IND-CCA2 security. This is contrast to the results in [12] for example, showing that there are encryption schemes secure in the random oracle model but which cannot be securely realized for any primitive, not even for a secure encryption scheme itself.

As for the existence of non-malleable pseudorandom generators, we are not aware if they can be derived from standard cryptographic assumptions, and we leave this as an interesting open problem. We also remark that, while non-malleability under chosen-image attacks seems to be a rather synthetic property, plain non-malleability as required in the NM-CPA result appears to be a modest and plausible assumption for typical instantiation candidates like hash functions. For instance, it should not be easy to flip bits in given hash value, affecting bits in the pre-image in a reasonable way.

FULL INSTANTIATION RESULT. Our main result is a standard-model security proof for a fully instantiated OAEP. It is not very reasonable to expect a proof of IND-CCA2 security of OAEP in the standard model, even assuming very strong properties of instantiating functions (although we all would like to see such result). As we mentioned above, we are not aware if one can even show IND-CPA security of fully instantiated OAEP.

Nevertheless we show that OAEP in the standard model can be proven to satisfy a rather strong notion of security notion, namely $\$$ NM-CPA. It is slightly weaker than the standard non-malleability notion NM-CPA in that there is a restriction that an unknown random message is encrypted in the challenge ciphertext. A bit more formally this security notion $\$$ NM-CPA requires that given a public key and a ciphertext of a challenge message chosen uniformly at random from a large message space it is hard to compute a valid ciphertext of a message non-trivially related to the challenge message. Note that this is consistent with how asymmetric schemes are typically used to build hybrid encryption schemes, where the key of the symmetric scheme is derived from a random string encrypted with the public-key scheme. To appreciate the power of the $\$$ NM-CPA definition we note that it implies for example the notion of OW-CPA and, moreover, Bleichenbacher's attack [7] on PKCS #1 v1.5 is not possible for $\$$ NM-CPA secure schemes.² Thus our result provides better evidence that OAEP resists such attacks, and specifies what properties of the instantiating functions are sufficient for this.

For our full instantiation proof we consider $\text{OAEP}^{G,H}[F_{\text{lsb}||\text{t-clear}}]$ where the t -part and the least significant bits of the s -part are output in clear. To achieve the $\$$ NM-CPA security notion under full instantiation of both oracles G and H in

² Bleichenbacher's attack works by generating a sequence of ciphertexts from a given ciphertext and verifying validity of the derived ciphertexts by querying the decryption oracle. While requiring *adaptive* queries to recover the entire message, one can view the message in first derived ciphertext in such an attack as having a small (but not negligible) probability of being non-trivially related to the original (possibly random) message.

OAEP^{G,H}[$F_{\text{lsb}||\text{t-clear}}$] we need to augment the near-collision resistant generator G by a trapdoor property, allowing to invert images efficiently given the trapdoor information; such generators exist if trapdoor permutations exist. We again use a non-malleable pseudorandom generator H for instantiating H . Assuming that the generators above exist we show that OAEP^{G,H}[$F_{\text{lsb}||\text{t-clear}}$] is $\$$ NM-CPA.³

To give further evidence of the usefulness of the $\$$ NM-CPA notion we finally show that we can derive a hybrid encryption scheme that is NM-CPA in the random oracle model from an asymmetric scheme secure in the sense of $\$$ NM-CPA. For this, one encrypts a random string r with the asymmetric scheme and then runs r through an idealized key derivation process to obtain $K = G(r)$, modeled through a random oracle G . The actual message is then encrypted with a symmetric scheme for key K . The construction of such hybrid encryption schemes resembles the encryption method in SSL/TLS [18]. There, simply speaking, the client encrypts a random string under the server’s public key and then both parties derive the actual symmetric key K by hashing the random string iteratively. If one considers this hashing step as an idealized process then our results provide a security guarantee for this technique. Observe that this result is still cast in the random oracle model; yet it separates the security of the key derivation process from the security of the asymmetric encryption scheme and can be seen as a partial instantiation for the random oracles in the encryption algorithm.

PROSPECT. The random oracle model should provide confidence that the design of a cryptographic scheme is sound, even if a security proof in the standard model for this scheme is missing. The heuristic argument is that “good” instantiations of random oracles then give evidence that no “clever” attacks against a scheme work. But the well-known negative results about the random oracle principle have raised some doubts how much confidence this security heuristic really gives.

The approach we take here towards challenging the doubts is to trade security goals against partial or full instantiations of random oracles. Our “test case” OAEP shows that this is a viable way and gives more insights in “how clever” attacks against the instantiations would have to be. And while this still does not rule out the possibility of extraordinary attacks we see this as an important supplement to the random oracle heuristic and to the question how instantiating candidates should be selected, hopefully inciting other results along this direction.

2 Preliminaries

If S is a set then $x \stackrel{\$}{\leftarrow} S$ means that the value x is chosen uniformly at random from S . If \mathcal{A} is a deterministic (resp. randomized algorithm) with a single output then $x \leftarrow \mathcal{A}(y, z, \dots)$ (resp. $x \stackrel{\$}{\leftarrow} \mathcal{A}(y, z, \dots)$) means that the value x is assigned the output of \mathcal{A} for input (y, z, \dots) . An algorithm is called efficient if it runs

³ Very recently, Brown [2] has shown that RSA-OAEP cannot be proven OW-CPA under certain security reductions. Our approach here does not fall under this kind of reductions and does not contradict his result. We provide more details in Section 3.2.

in polynomial time in the input length (which, in our case, usually refers to polynomial time in the security parameter).

A function family $F = \bigcup_k F(1^k)$ consists of sets of functions $F(1^k) = \{f : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{n(k)}\}$. It is called a family of trapdoor permutations if for each $f \in F(1^k)$ there exists f^{-1} such that $f(f^{-1}) \equiv \text{ID}$. We usually identify the functions f and f^{-1} simply with their descriptions, and write $(f, f^{-1}) \stackrel{\$}{\leftarrow} F(1^k)$ for the random choice of f (specifying also f^{-1}) from the family $F(1^k)$. Unless stated differently the minimal assumption about a function family in this paper is that it is one-way, and that it is efficiently computable.

2.1 The OAEP Framework

The OAEP encryption framework [5] is parameterized by integers k, k_0 and k_1 (where k_0, k_1 are linear functions of k) and makes use of a trapdoor permutation family F with domain and range $\{0, 1\}^k$ and two random oracles

$$G: \{0, 1\}^{k_0} \rightarrow \{0, 1\}^{k-k_0} \quad \text{and} \quad H: \{0, 1\}^{k-k_0} \rightarrow \{0, 1\}^{k_0}.$$

The message space is $\{0, 1\}^{k-k_0-k_1}$. The scheme $\text{OAEP}^{G,H}[F] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined as follows:

- The key generation algorithm $\mathcal{K}(1^k)$ picks a pair $(f, f^{-1}) \leftarrow F(1^k)$ at random. Let pk specify f and let sk specify f^{-1} .
- The encryption algorithm $\mathcal{E}(pk, M)$ picks $r \stackrel{\$}{\leftarrow} \{0, 1\}^{k_0}$, and computes $s \leftarrow G(r) \oplus (M || 0^{k_1})$ and $t \leftarrow H(s) \oplus r$. It finally outputs $C \leftarrow f(s || t)$.
- The decryption algorithm $\mathcal{D}(sk, C)$ computes $s || t \leftarrow f^{-1}(C)$, $r \leftarrow t \oplus H(s)$ and $M \leftarrow s \oplus G(r)$. If the last k_1 bits of M are zeros, then it returns the first $k - k_0 - k_1$ bits of M , else it returns \perp .

The encryption scheme $\text{OAEP}^{G,H}[F]$ is IND-CCA2 secure in the RO model if the underlying trapdoor permutation family F is partial one-way [16].

As a side effect of the partial one-wayness result for OAEP [16] we can immediately conclude security of a particular OAEP variant, where we use partial one-way trapdoor permutation family $F_{\text{t-clear}}$ based on a trapdoor permutation function family F . Namely, each function $f_{\text{t-clear}} : \{0, 1\}^k \rightarrow \{0, 1\}^k$ in $F_{\text{t-clear}}$ is described by $f_{\text{t-clear}}(s || t) \equiv f(s) || \text{ID}(t) = f(s) || t$ for a one-way permutation $f : \{0, 1\}^{k-k_0} \rightarrow \{0, 1\}^{k-k_0}$, i.e., the t -part is output in clear. A random instance $(f_{\text{t-clear}}, f_{\text{t-clear}}^{-1}) \leftarrow F_{\text{t-clear}}(1^k)$ is sampled by picking $(f, f^{-1}) \leftarrow F(1^k)$ and setting $f_{\text{t-clear}}$ as above (the inverse $f_{\text{t-clear}}^{-1}$ is straightforwardly defined). Then $F_{\text{t-clear}}$ is clearly partial one-way and thus $\text{OAEP}^{G,H}[F_{\text{t-clear}}]$ IND-CCA2 secure in the random oracle model.

Analogously, we consider another important variant of OAEP where we also output the k_1 least significant bits $\text{lsb}_{k_1}(s)$ of s in clear and merely apply the trapdoor function f to the leading $k - k_0 - k_1$ bits of s . That is, a random function $f_{\text{lsb} || \text{t-clear}} : \{0, 1\}^k \rightarrow \{0, 1\}^k$ in $F_{\text{lsb} || \text{t-clear}}(1^k)$ is described by a random trapdoor permutation $f : \{0, 1\}^{k-k_0-k_1} \rightarrow \{0, 1\}^{k-k_0-k_1}$ and $f_{\text{lsb} || \text{t-clear}}(s || t) =$

$f(s_{1\dots k-k_0-k_1})||\text{lsb}_{k_1}(s)||t$. Note that since $s = G(r) \oplus M||0^{k_1}$ this means that we output the least significant bits $\text{lsb}_{k_1}(G(r))$ of $G(r)$ and t in clear. For this reason we sometimes write $s||\gamma$ instead of s and denote by γ the k_1 bits $\text{lsb}_{k_1}(G(r))$ such that $f_{|\text{lsb}||t\text{-clear}}(s||\gamma||t) = f(s)||\gamma||t$. $F_{|\text{lsb}||t\text{-clear}}$ is clearly partial one-way and $\text{OAEP}^{G,H}[F_{|\text{lsb}||t\text{-clear}}]$ is IND-CCA2 secure in the random oracle model.

In both cases we often identify $F_{t\text{-clear}}$ resp. $F_{|\text{lsb}||t\text{-clear}}$ simply with the underlying family F and vice versa. In particular we often denote a random function from $F_{t\text{-clear}}$ or $F_{|\text{lsb}||t\text{-clear}}$ simply by f . We call $F_{t\text{-clear}}$ resp. $F_{|\text{lsb}||t\text{-clear}}$ *the induced family of F* .

RANDOM ORACLE INSTANTIATIONS. For an instantiation of the random oracle G in $\text{OAEP}^{G,H}[F]$ we consider a pair of efficient algorithms $\mathcal{G} = (\text{KGenG}, \text{G})$ where KGenG on input 1^k returns a random key K and the deterministic algorithm⁴ G maps this key K and input $r \in \{0, 1\}^{k_0}$ to an output string $\text{G}(K, r) = \text{G}_K(r)$ of $k - k_0$ bits. Then we write $\text{OAEP}^{\mathcal{G},H}[F]$ for the encryption scheme which works as defined above, but where the key pair (sk, pk) is now given by $sk = (f^{-1}, K)$ and $pk = (f, K)$ and where each evaluation of $G(r)$ is replaced by $\text{G}_K(r)$. We say that $\text{OAEP}^{\mathcal{G},H}[F]$ is a *partial G -instantiation of OAEP through \mathcal{G}* .

A *partial H -instantiation $\text{OAEP}^{G,\mathcal{H}}[F]$ of OAEP through \mathcal{H}* and partial instantiations of the aforementioned OAEP variations are defined accordingly. If we instantiate both oracles G, H simultaneously then we speak of a *full instantiation $\text{OAEP}^{\mathcal{G},\mathcal{H}}[F]$ of OAEP through \mathcal{G} and \mathcal{H}* .

2.2 Security of Encryption Schemes

In this section we review the relevant security notions for asymmetric encryption schemes $\text{AS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. In addition to indistinguishability under chosen-plaintext and chosen-ciphertext attacks (IND-CPA, IND-CCA1, IND-CCA2) — see for instance [3] for formal definitions— we occasionally also rely on the notions of non-malleability. This notion was introduced and formalized in [15,3]. The most basic version is called NM-CPA and says that a ciphertext of a message M^* should not help to find a ciphertext of a related message M , where the distribution of message M^* is defined by an efficient distribution \mathcal{M} and related messages are specified by an efficient relation R , both chosen by the adversary.

Definition 1 (NM-CPA). *Let AS be an asymmetric encryption scheme. Then AS is called secure in the sense of NM-CPA if for every efficient algorithm A the following random variables $\text{Exp}_{\text{AS},A}^{\text{nm-cpa-1}}(k)$, $\text{Exp}_{\text{AS},A}^{\text{nm-cpa-0}}(k)$ are computationally indistinguishable:*

⁴ In general, the instantiating functions can be randomized. This requires some care with the decryption algorithms and possibly introduces new attacks. Since our results all hold with respect to deterministic algorithms this is beyond our scope here; see [9] for more details.

<p>Experiment $\text{Exp}_{\text{AS},\mathcal{A}}^{\text{nm-cpa-1}}(k)$</p> <p>$(pk, sk) \xleftarrow{\\$} \mathcal{K}(1^k)$</p> <p>$(\mathcal{M}, \text{state}) \xleftarrow{\\$} \mathcal{A}(pk)$</p> <p>$M^* \xleftarrow{\\$} \mathcal{M}$</p> <p>$C^* \xleftarrow{\\$} \mathcal{E}_{pk}(M^*)$</p> <p>$(R, C) \xleftarrow{\\$} \mathcal{A}(\text{state}, C^*)$</p> <p>$M \leftarrow \mathcal{D}_{sk}(C)$</p> <p>Return 1 iff</p> <p style="text-align: center;">$(C \neq C^*) \wedge R(M^*, M)$</p>	<p>Experiment $\text{Exp}_{\text{AS},\mathcal{A}}^{\text{nm-cpa-0}}(k)$</p> <p>$(pk, sk) \xleftarrow{\\$} \mathcal{K}(1^k)$</p> <p>$(\mathcal{M}, \text{state}) \xleftarrow{\\$} \mathcal{A}(pk)$</p> <p>$M^* \xleftarrow{\\$} \mathcal{M}; M' \xleftarrow{\\$} \mathcal{M}$</p> <p>$C' \xleftarrow{\\$} \mathcal{E}_{pk}(M')$</p> <p>$(R, C) \xleftarrow{\\$} \mathcal{A}(\text{state}, C')$</p> <p>$M \leftarrow \mathcal{D}_{sk}(C)$</p> <p>Return 1 iff</p> <p style="text-align: center;">$(C \neq C') \wedge R(M^*, M)$</p>
---	--

It is assumed that the messages in the support of \mathcal{M} have equal length.

We note that the original definition of NM-CPA in [3] actually allows the adversary to output a vector of ciphertexts. Our results for OAEP merely hold with respect to binary relations and therefore we restrict the definition here to such relations. We remark that the aforementioned relationships of NM-CPA to the indistinguishability notions, e.g., that this notion is strictly stronger than the one of IND-CPA, hold for relations of arity two as well.

We define a weaker security notion is that of \$NM-CPA where the adversary does not have the ability to choose a distribution over the messages, but where a random message is encrypted and the adversary tries to find a ciphertext of a related message.

Definition 2 (\$NM-CPA). *Let $\text{AS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme and let \mathcal{M} for input 1^k describe the uniform distribution over all $\ell(k)$ bit strings for some polynomial ℓ . Then AS is called secure in the sense of \$NM-CPA if for every efficient algorithm \mathcal{A} and for every efficient relation R the following random variables $\text{Exp}_{\text{AS},\mathcal{A},\mathcal{M},R}^{\text{nm-cpa-1}}(k)$, $\text{Exp}_{\text{AS},\mathcal{A},\mathcal{M},R}^{\text{nm-cpa-0}}(k)$ are computationally indistinguishable:*

<p>Experiment $\text{Exp}_{\text{AS},\mathcal{A},\mathcal{M},R}^{\text{nm-cpa-1}}(k)$</p> <p>$(pk, sk) \xleftarrow{\\$} \mathcal{K}(1^k)$</p> <p>$M^* \xleftarrow{\\$} \mathcal{M}(1^k)$</p> <p>$C^* \xleftarrow{\\$} \mathcal{E}_{pk}(M^*)$</p> <p>$C \xleftarrow{\\$} \mathcal{A}(pk, C^*, \langle R \rangle)$</p> <p>$M \leftarrow \mathcal{D}_{sk}(C)$</p> <p>Return 1 iff</p> <p style="text-align: center;">$(C \neq C^*) \wedge R(M^*, M)$</p>	<p>Experiment $\text{Exp}_{\text{AS},\mathcal{A},\mathcal{M},R}^{\text{nm-cpa-0}}(k)$</p> <p>$(pk, sk) \xleftarrow{\\$} \mathcal{K}(1^k)$</p> <p>$M^* \xleftarrow{\\$} \mathcal{M}(1^k); M' \xleftarrow{\\$} \mathcal{M}(1^k)$</p> <p>$C' \xleftarrow{\\$} \mathcal{E}_{pk}(M')$</p> <p>$C \xleftarrow{\\$} \mathcal{A}(pk, C', \langle R \rangle)$</p> <p>$M \leftarrow \mathcal{D}_{sk}(C)$</p> <p>Return 1 iff</p> <p style="text-align: center;">$(C \neq C') \wedge R(M^*, M)$</p>
---	---

While the notion of \$NM-CPA is weaker than the one of NM-CPA —in addition to the restriction to uniformly distributed messages the relation is now fixed in advance— it yet suffices for example to show security in the sense of OW-CPA (where the adversary’s goal is to recover a random message in a given ciphertext) and it also covers Bleichenbacher’s attack on PKCS #1 v1.5. In Section 5 we also show that the notion of \$NM-CPA is enough to derive NM-CPA security under an idealized key derivation function. Namely, one encrypts a random

string r under the $\$$ NM-CPA public-key encryption scheme and then pipes r through a random oracle G to derive a key $K = G(r)$ for the symmetric scheme. In fact, one can view the SSL encryption method where the client sends an encrypted random key to the server and both parties derive a symmetric key through a complicated hash function operation as a special case of this method. Then this result about lifting $\$$ NM-CPA to NM-CPA security, together with the $\$$ NM-CPA security proof for the full instantiation of $\text{OAEP}_{|\text{sb}||\text{t-clear}}$, provides an interesting security heuristic (as long as the key derivation process behaves in an ideal way).

2.3 Pseudorandom Generators

Typically, the minimal expected requirement when instantiating a random oracle is that the instantiating function describes a pseudorandom generator, consisting of the key generation algorithm KGen producing a public key K and the evaluation algorithm G mapping a random seed r with key K to the pseudorandom output. Usually the output of this generator should still look random when some side information $\text{hint}(r)$ about the seed r is given. This probabilistic function hint must be of course uninvertible, a weaker notion than one-wayness (cf. [11]).

We also incorporate into the definition the possibility that the key generation algorithm outputs some secret trapdoor information K^{-1} in addition to K . Given this information K^{-1} one can efficiently invert images. If this trapdoor property is not required we can assume that $K^{-1} = \perp$ and often omit K^{-1} in the key generator's output.

Definition 3 ((Trapdoor) Pseudorandom Generator). *Let KGen be an efficient key-generation algorithm that takes as input 1^k for $k \in \mathbb{N}$ and outputs a key K ; let G be an efficient deterministic evaluation algorithm that, on input K and a string $r \in \{0, 1\}^k$ returns a string of length $\ell(k)$. Then $\mathcal{G} = (\text{KGen}, G)$ is called a pseudorandom generator (with respect to hint) if the following random variables are computationally indistinguishable:*

- Let $K \leftarrow \text{KGen}(1^k)$, $r \xleftarrow{\$} \{0, 1\}^k$, $h \leftarrow \text{hint}(r)$, output $(K, G(K, r), h)$.
- Let $K \leftarrow \text{KGen}(1^k)$, $r \xleftarrow{\$} \{0, 1\}^k$, $h \leftarrow \text{hint}(r)$, $u \leftarrow \{0, 1\}^{\ell(n)}$, output (K, u, h) .

Furthermore, if there is an efficient algorithm TdG such that for any $k \in \mathbb{N}$, any $(K, K^{-1}) \leftarrow \text{KGen}(1^k)$, any $r \in \{0, 1\}^k$ we have $G(K, \text{TdG}(K^{-1}, G(K, r))) = G(K, r)$ then $(\text{KGen}, G, \text{TdG})$ is called a trapdoor pseudorandom generator.

For our results about OAEP we often need further properties from the pseudorandom generator, including near-collision resistance and non-malleability. The former means that given a seed r it is hard to find a different seed r' such that $G(K, r)$ and $G(K, r')$ coincide on a predetermined set of bits (even if they are allowed to differ on the other bits). Non-malleability refers to generators where the generator's output for a seed should not help to produce an image of a related seed. We give precise definitions and details concerning existential questions on site.

3 Partial Instantiations for OAEP

In this section we prove security of partial instantiations of OAEP. Our results show that one can replace either one of the random oracle in OAEP by reasonable primitives and still maintain security (in the random oracle model).

3.1 Instantiating the G -Oracle for IND-CCA2 Security

We first show how to construct a pseudorandom generator with a special form of collision-resistance. This property says that finding an input r' to a random input r , such that $G(K, r)$ and $G(K, r')$ coincide on the k least significant bits $\text{lsb}_k(G(K, r)), \text{lsb}_k(G(K, r'))$, is infeasible. According to comparable collision types for hash functions [6] we call this *near-collision resistance*.

Definition 4 (Near-collision Resistant Pseudorandom Generator). *A pseudorandom generator $\mathcal{G} = (\text{KGen}, G)$ is called near-collision resistant (for the least significant k bits) if for any efficient algorithm \mathcal{C} the following holds: Let $K \leftarrow \text{KGen}(1^k)$, $r \leftarrow \{0, 1\}^k$, $r' \leftarrow \mathcal{C}(K, r)$. Then the probability that $r \neq r'$ but $\text{lsb}_k(G(K, r)) = \text{lsb}_k(G(K, r'))$ is negligible.*

Near-collision resistant generators can be built, for example, from one-way permutations via the well-known Yao-Blum-Micali construction [22,8]. In that case, given a family G of one-way permutations the key generation algorithm $\text{KGen}_{\text{YBM}}(1^k)$ of this generator simply picks a random instance $g : \{0, 1\}^k \rightarrow \{0, 1\}^k$ of $G(1^k)$, and $G_{\text{YBM}}(g, r) = (\text{hb}(r), \text{hb}(g(r)), \dots, \text{hb}(g^{n-1}(r)), g^n(r))$ is defined through the hardcore bits hb of g . Since g is a permutation different inputs $r \neq r'$ yield different output parts $g^n(r) \neq g^n(r')$.

Given a near-collision resistant pseudorandom generator we show how to instantiate the G -oracle in $\text{OAEP}^{\mathcal{G}, H}[F_{\text{t-clear}}]$ for the family $F_{\text{t-clear}}$ which is induced by a trapdoor permutation family F (i.e., where a member $f : \{0, 1\}^{k-k_0} \rightarrow \{0, 1\}^{k-k_0}$ of F is applied to the k -bit inputs such that the lower k_0 bits are output in clear).

Theorem 1. *Let $\mathcal{G} = (\text{KGen}, G)$ be a pseudorandom generator which is near-collision resistant (for the k_1 least significant bits). Let F be trapdoor permutation family and let $F_{\text{t-clear}}$ be the induced partial one-way trapdoor permutation family defined in Section 2.1. Then the partial G -instantiation $\text{OAEP}^{\mathcal{G}, H}[F_{\text{t-clear}}]$ of OAEP through \mathcal{G} is IND-CCA2 in the random oracle model.*

The full proof appears in the full version [3]. The idea is to gradually change the way the challenge ciphertext (encrypting one of two adversarially chosen messages, the hidden choice made at random) is computed in a sequence of games. We show that each of these steps does not change an adversary’s success probability of predicting the secret choice noticeably:

- Initially, in Game^0 the challenge ciphertext $f(s^*)||t^*$ for message M^* is computed as in the scheme’s description by $s^* = G(K, r^*) \oplus M^*||0^{k_1}$ for the near-collision resistant generator G and $t^* = H(s^*) \oplus r^*$ for oracle H .

- In **Game**¹ the ciphertext is now computed by setting $s^* = G(K, r^*) \oplus M^* || 0^{k_1}$ as before, but letting $t^* = \omega \oplus r^*$ for a random ω which is independent of $H(s^*)$. Because H is a random oracle this will not affect the adversary’s success probability, except for the rare case that the adversary queries H about s^* .
- In **Game**², in a rather cosmetic change, we further substitute $t^* = \omega \oplus r^*$ simply for $t^* = \omega$, making the t -part independent of the generator’s pre-image r^* .
- in **Game**³ we use the pseudorandomness of generator G to replace $s^* = G(K, r^*) \oplus M^* || 0^{k_1}$ by $s^* = u \oplus M^* || 0^{k_1}$ for a random u .

Since ciphertexts in the last game are distributed independently of the actual message security of the original scheme follows, after a careful analysis that decryption queries do not help; this is the step where we exploit that H is still a random oracle and that \mathcal{G} is near-collision resistant. Namely, the near-collision resistance prevents an adversary from transforming the challenge ciphertext for values r^*, s^* into a valid one for the same s^* but a different r ; otherwise the least significant bits of $s^* = G(K, r^*) \oplus M^* || 0^{k_1} = G(K, r) \oplus M || 0^{k_1}$ would not coincide and the derived ciphertext would be invalid with high probability. Given this, the adversary must always use a “fresh” value s when submitting a ciphertext to the decryption oracle, and must have queried the random oracle H about s before (or else the ciphertext is most likely invalid). But then the adversary already “knows” $r = t \oplus H(s)$ —recall that for $F_{t\text{-clear}}$ the t -part is included in clear in ciphertexts— and therefore “knows” the (padded) message $M || z = s \oplus G(K, r)$ encapsulated in the ciphertext.

3.2 Instantiating the H -Oracle

To instantiate the H -oracle we introduce the notion of a non-malleable pseudorandom generator. For such a pseudorandom generator it should be infeasible to find for a given image $y^* = H_K(s^*)$ of a random s^* a different image $y = H_K(s)$ of a related value s , where the corresponding efficient relation $R(s^*, s)$ must be determined *before* seeing K and y^* .⁵ More precisely, we formalize non-malleability of a pseudorandom generator by the indistinguishability of two experiments. For any adversary \mathcal{B} it should not matter whether \mathcal{B} is given $f(s^*), y^* = H_K(s^*)$ or $f(s'), y' = H_K(s')$ for an independent s' instead: the probability that \mathcal{B} outputs $f(s)$ and $y = H_K(s)$ such that s is related to s^* via relation R should be roughly the same in both cases.⁶

⁵ We are thankful to the people from the ECRYPT network for pointing out that a possibly stronger definition for adaptively chosen relations allows trivial relations over the images and cannot be satisfied.

⁶ Adding the image under the trapdoor permutation uniquely determines the pre-image of the pseudorandom generator’s output and enables us to specify $R(s^*, s)$ via *the* pre-images. Since this also bundles the security of the trapdoor permutation and the generator, Brown’s recent impossibility result about security reductions for OAEP [2] does not apply.

Definition 5 (Non-Malleable Pseudorandom Generator). Assume $\mathcal{H} = (\text{KGenH}, \text{H})$ is a pseudorandom generator (which is pseudorandom with respect to $\text{hint}(x) = (f, f(x))$ for $(f, f^{-1}) \leftarrow F(1^k)$ from the trapdoor function family F). Then \mathcal{H} is called non-malleable with respect to hint if for any efficient algorithm \mathcal{B} and any efficient relation R the following random variables $\text{Exp}_{\mathcal{H}, \mathcal{B}, F, R}^{\text{nm-cpa-1}}(k)$, $\text{Exp}_{\mathcal{H}, \mathcal{B}, F, R}^{\text{nm-cpa-0}}(k)$ are computationally indistinguishable, where the experiments are defined as follows.

Experiment $\text{Exp}_{G, \mathcal{B}, F, R}^{\text{nm-cpa-1}}(k)$

$K \xleftarrow{\$} \text{KGenH}(1^k)$
 $(f, f^{-1}) \xleftarrow{\$} F$
 $s^* \xleftarrow{\$} \{0, 1\}^k$
 $y^* \xleftarrow{\$} \text{H}_K(s^*)$
 $(z, y) \xleftarrow{\$} \mathcal{B}(K, f, f(s^*), y^*)$
 $s \leftarrow f^{-1}(z)$
 Return 1 iff
 $R(s^*, s) \wedge \text{H}_K(s) = y \wedge s^* \neq s$

Experiment $\text{Exp}_{G, \mathcal{B}, F, R}^{\text{nm-cpa-0}}(k)$

$K \xleftarrow{\$} \text{KGenH}(1^k)$
 $(f, f^{-1}) \xleftarrow{\$} F$
 $s^* \xleftarrow{\$} \{0, 1\}^k ; s' \xleftarrow{\$} \{0, 1\}^k$
 $y' \xleftarrow{\$} \text{H}_K(s')$
 $(z, y) \xleftarrow{\$} \mathcal{B}(K, f, f(s^*), y')$
 $s \leftarrow f^{-1}(z)$
 Return 1 iff
 $R(s^*, s) \wedge \text{H}_K(s) = y \wedge s^* \neq s$

Given a non-malleable pseudorandom generator we can prove NM-CPA security of the partial H -instantiation of OAEP, under the restriction that the adversarial chosen message distribution and relation are defined at the beginning of the attack via $(\mathcal{M}, R, \text{state}) \leftarrow \mathcal{A}(1^k)$ and thus depend only the security parameter. This relaxed notion still implies for example IND-CPA security (but for messages picked independently of the public key), is still incomparable to IND-CCA1 security, and also thwarts Bleichenbacher’s attack. We call such schemes *NM-CPA* for pre-defined message distributions and relations.

Theorem 2. Let F be a trapdoor permutation family and let $F_{\text{t-clear}}$ be the induced partial one-way trapdoor permutation family. Let $\mathcal{H} = (\text{KGenH}, \text{H})$ be a pseudorandom generator (with respect to $\text{hint}(x) = (f, f(x))$ for $(f, f^{-1}) \leftarrow F(1^k)$). Assume further that \mathcal{H} is non-malleable with respect to hint . Then the partial H -instantiation $\text{OAEP}^{G, \mathcal{H}}[F_{\text{t-clear}}]$ through \mathcal{H} is NM-CPA for pre-defined message distributions and relations in the random oracle model.

The proof idea is as follows. Assume that an attacker, given a ciphertext for some values r^*, s^* (which uniquely define the message in a ciphertext), tries to prepare a related ciphertext for some value $r \neq r^*$, without having queried random oracle G about r before. Then such a ciphertext is most likely invalid because with overwhelming probability the least significant bits of $s \oplus G(r)$ are not zero. Else, if $r = r^*$, then we must have $f(s) \neq f(s^*)$ and $s \neq s^*$, since the adversarial ciphertext must be different for a successful attack. But then the values $\text{H}(K, s^*)$ and $\text{H}(K, s)$ for different pre-images must be related via the ciphertext’s relation, contradicting the non-malleability of the generator H . In any other case, if $r \neq r^*$ and r is among the queries to G , the random value $G(r^*)$ is independent of $G(r)$. So must be the messages $M^* || 0^{k_1} = s^* \oplus G(r^*)$ and $M || 0^{k_1} = s \oplus G(r)$, as required for non-malleability. Details can be found in the full version [3].

Replacing the H -oracle without violating IND-CCA2 security is more ambitious and we require a very strong assumption on the pseudorandom generator, called non-malleability under chosen-image attacks (where the adversary can also make inversion queries to the trapdoor pseudorandom generator). Since any pseudorandom generator with this property is already close to a chosen-ciphertext secure encryption scheme, we rather see this as an indication that a partial instantiation might be possible and that separation results as [12,19,20,1,17,21,9,14] seem to be hard to find. The formal treatment of the following and the proof appear in the full version [10].

Theorem 3. *Let F be trapdoor permutation family and let $F_{t\text{-clear}}$ be the induced partial one-way trapdoor permutation family defined in Section 2.1. Let $\mathcal{H} = (\text{KGEnH}, H, \text{TdH})$ be a trapdoor pseudorandom generator which is non-malleable under chosen-image attacks (with respect to $\text{hint}(x) = (f, f(x))$ for $(f, f^{-1}) \leftarrow F_{t\text{-clear}}(1^k)$). Then the partial H -instantiation $\text{OAEP}^{\mathcal{G}, \mathcal{H}}[F_{t\text{-clear}}]$ through \mathcal{H} is IND-CCA2 in the random oracle model.*

4 Full Instantiation for OAEP

In this section we prove that there exists a full instantiation of $\text{OAEP}_{\text{lsb}||\text{t-clear}}$ which is secure in the sense of $\$$ NM-CPA in the standard model, implying for example that the scheme is OW-CPA. Recall that in $\text{OAEP}_{\text{lsb}||\text{t-clear}}$ we write $s||\gamma = G(s) \oplus M||0^{k_1}$ instead of s to name the least significant bits explicitly.

To prove our result we need a near-collision resistant *trapdoor* pseudorandom generator, i.e., which combines near-collision resistance with the trapdoor property. Such generators can be easily built by using again the Blum-Micali-Yao generator, but this time by deploying a trapdoor permutation g instead of a one-way permutation, i.e., the generator's output for random r is given by $\text{G}_{\text{YBM}}(g, r) = (\text{hb}(r), \text{hb}(g(r)), \dots, \text{hb}(g^{n-1}(r)), g^n(r))$. Letting K^{-1} contain the trapdoor information g^{-1} algorithm TdG can easily invert the k_1 least significant bits y of the output to recover a pre-image r .

To be precise we make use of two additional, specific properties of the Blum-Micali-Yao generator. First, we assume that recovering a pre-image is possible given the k_1 least significant bits only, i.e., without seeing the remaining part of the image. To simplify the proof we furthermore presume that the k_1 least significant bits of the generator's output are statistically close to uniform (over the choice of the seed).⁷ We simply refer to generators with the above properties as a *near-collision resistant trapdoor pseudorandom generator (for the least significant k bits)*.

Theorem 4. *Let F be trapdoor permutation family and let $F_{\text{lsb}||\text{t-clear}}$ be the induced partial one-way trapdoor permutation family. Let $\mathcal{G} = (\text{KGenG}, G)$ be a*

⁷ It is easy to adapt the proof to the more general case of arbitrary distributions of the least significant bits, as long as they support extraction. But this would also require to change the definition of the non-malleable pseudorandom generator $\text{G}_{\text{KG}}(s||\gamma)$ to support arbitrary distributions on the γ -part.

near-collision resistant trapdoor pseudorandom generator (for the k_1 least significant bits). Let $\mathcal{H} = (\text{KGenH}, \text{H})$ be a generator which is pseudorandom and non-malleable with respect to $\text{hint}(s||\gamma) = (f, f(s)||\gamma)$ for $(f, f^{-1}) \leftarrow F(1^k)$. Then the full instantiation $\text{OAEP}^{\mathcal{G}, \mathcal{H}}_{[F]_{\text{sb}}||t\text{-clear}}$ through \mathcal{G} and \mathcal{H} is $\$NM\text{-CPA}$.

The proof appears in the full version [10]. The basic idea is similar to the one of NM-CPA security for the partial H -instantiation. The important difference is that the randomness of the encrypted message M in a ciphertext $f(s)||\gamma||t$ for $s||\gamma = \mathbf{G}_K(r) \oplus M||0^{k_1}$ helps to overcome otherwise existing ‘‘circular’’ dependencies between \mathcal{G} and \mathcal{H} in the computations of ciphertexts (which, in the partial instantiation case, do not occur due to the fact that G is a random oracle).

5 Hybrid Encryption from $\$NM\text{-CPA}$ Schemes

We show that a public-key scheme which is secure in the sense of $\$NM\text{-CPA}$ (i.e., for pre-defined relations), together with an IND-CCA2 secure symmetric scheme suffices to build a NM-CPA secure hybrid scheme in the random oracle model (i.e., even for adaptively chosen message distributions and relations).

Construction 1. Let $\text{AS} = (\mathcal{EK}_{\text{asym}}, \mathcal{E}_{\text{asym}}, \mathcal{D}_{\text{asym}})$ be an asymmetric encryption scheme and let $\text{SS} = (\mathcal{EK}_{\text{sym}}, \mathcal{E}_{\text{sym}}, \mathcal{D}_{\text{sym}})$ be a symmetric encryption scheme. Let G be a hash function mapping k -bit strings into the key space of the symmetric scheme. Then the hybrid encryption scheme $\text{AS}' = (\mathcal{EK}'_{\text{asym}}, \mathcal{E}'_{\text{asym}}, \mathcal{D}'_{\text{asym}})$ is defined as follows.

- The key generation algorithm $\mathcal{EK}'_{\text{asym}}(1^k)$ outputs a key pair $(\text{sk}, \text{pk}) \stackrel{\$}{\leftarrow} \mathcal{EK}_{\text{asym}}(1^k)$.
- The encryption algorithm $\mathcal{E}'_{\text{asym}}$ on input pk, M picks $r \stackrel{\$}{\leftarrow} \{0, 1\}^k$, computes $C_{\text{asym}} \stackrel{\$}{\leftarrow} \mathcal{E}_{\text{asym}}(\text{pk}, r)$, $C_{\text{sym}} \stackrel{\$}{\leftarrow} \mathcal{E}_{\text{sym}}(G(r), M)$ and returns $(C_{\text{asym}}, C_{\text{sym}})$.
- The decryption algorithm $\mathcal{D}'_{\text{asym}}$ on input $(C_{\text{asym}}, C_{\text{sym}})$ and sk computes $r \leftarrow \mathcal{D}_{\text{asym}}(\text{sk}, C_{\text{asym}})$, $M \leftarrow \mathcal{D}_{\text{sym}}(G(r), C_{\text{sym}})$ and returns M .

Theorem 5. Let $\text{AS} = (\mathcal{EK}_{\text{asym}}, \mathcal{E}_{\text{asym}}, \mathcal{D}_{\text{asym}})$ be an asymmetric encryption scheme which is $\$NM\text{-CPA}$. Let $\text{SS} = (\mathcal{EK}_{\text{sym}}, \mathcal{E}_{\text{sym}}, \mathcal{D}_{\text{sym}})$ be an IND-CCA2 symmetric encryption scheme. Let G be a hash function and assume $\text{AS}' = (\mathcal{EK}'_{\text{asym}}, \mathcal{E}'_{\text{asym}}, \mathcal{D}'_{\text{asym}})$ is the hybrid encryption scheme defined according to Construction 1. Then AS' is NM-CPA secure in the random oracle model.

The proof is in the full version [10] and actually shows that the scheme is NM-CPA with respect to the stronger notion where the adversary outputs a sequence $\mathbf{C} = (C_1, \dots, C_m)$ of ciphertexts and the success is measured according to $R(M^*, \mathbf{M})$ for $\mathbf{M} = (M_1, \dots, M_m)$.

Acknowledgments

We thank the anonymous reviewers for comments. Part of the work done while both authors were visiting Centre de Recerca Matemàtica (CRM) and Technical

University of Catalonia (UPC), Barcelona, Spain, whose support is highly appreciated. The second author was also supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

References

1. M. Bellare, A. Boldyreva and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Eurocrypt 2004*, Volume 3027 of *LNCS*, pp. 171–188. Springer-Verlag, 2004.
2. D. R. L. Brown. Unprovable Security of RSA-OAEP in the Standard Model. *Cryptology ePrint Archive, Report 2006/223*, 2006.
3. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO '98*, Volume 1462 of *LNCS*, pp. 26–45. Springer-Verlag, 1998.
4. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93*, pp. 62–73. ACM, 1993.
5. M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. In *Eurocrypt '94*, Volume 950 of *LNCS*, pp. 92–111. Springer-Verlag, 1995.
6. E. Biham and R. Chen. Near-Collisions of SHA-0. In *CRYPTO '2004*, Volume 3152 of *LNCS*, pp. 290–305. Springer-Verlag, 2004.
7. D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *CRYPTO '98*, Volume 1462 of *LNCS*, pp. 1–12. Springer-Verlag, 1998.
8. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *Journal on Computing*, Volume 13, pp. 850–864, SIAM, 1984.
9. A. Boldyreva and M. Fischlin. Analysis of random-oracle instantiation scenarios for OAEP and other practical schemes. In *CRYPTO 2005*, Volume 3621 of *LNCS*, pp. 412–429. Springer-Verlag, 2005.
10. A. Boldyreva and M. Fischlin. On the Security of OAEP. Full version of this paper, available from the authors' homepages. 2006.
11. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO '97*, Volume 1294 of *LNCS*. pp. 455–469. Springer-Verlag, 1997.
12. R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. In *STOC '98*, pp. 209–218. ACM, 1998.
13. R. Canetti, D. Micciancio and O. Reingold. Perfectly one-way probabilistic hash functions. In *STOC '98*, pp. 131–140. ACM, 1998.
14. Y. Dodis, R. Oliveira, and K. Pietrzak. On the generic insecurity of full-domain hash. In *CRYPTO 2005*, Volume 3621 of *LNCS*, pp. 449–466. Springer-Verlag, 2005.
15. D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. *Journal on Computing*, Vol. 30(2), pp. 391–437. SIAM, 2000.
16. E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA-OAEP is secure under the RSA assumption. In *CRYPTO 2001*, volume 2139 of *LNCS*, pp. 260–274. Springer-Verlag, 2001.
17. S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS 2003*. IEEE, 2003.
18. IETF-TLS Working Group. Transport Layer Security. <http://www.ietf.org/html.charters/tls-charter.html>, November 2005.

19. U. Maurer, R. Renner and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *TCC 2004*, volume 2951 of *LNCS*, pp. 21–39. Springer-Verlag, 2004.
20. J. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO 2002*, volume 2442 of *LNCS*, pp. 111–126. Springer-Verlag, 2002.
21. P. Paillier and D. Vergnaud. Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log. In *Asiacrypt 2005*, volume 3788 of *LNCS*, pp. 1–20. Springer-Verlag, 2005.
22. A. Yao. Theory and applications of trapdoor functions. In *FOCS '82*, pp. 80–91. IEEE, 1982.

Relationship Between Standard Model Plaintext Awareness and Message Hiding

Isamu Teranishi^{1,2} and Wakaha Ogata²

¹ NEC Corporation,
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa, 211-8666, Japan
² Tokyo Institute of Technology,
2-12-1 Ookayama, Meguro-ku Tokyo, 152-8550, Japan
teranisi@ah.jp.nec.com, wakaha@mot.titech.ac.jp

Abstract. Recently, Bellare and Palacio succeeded in defining the plaintext awareness, which is also called PA2, in the standard model. They propose three variants of the standard model PA2 named perfect, statistical, and computational PA2. In this paper, we study the relationship between the standard model PA2 and the property about message hiding, that is, IND-CPA. Although it seems that these two are independent notions at first glance, we show that all of the perfect, statistical, and computational PA2 in the standard model imply the IND-CPA security if the encryption function is oneway. By using this result, we also showed that “PA2 + Oneway \Rightarrow IND-CCA2”. This result shows the “all-or-nothing” aspect of the PA2. That is, a standard model PA2 secure public-key encryption scheme either satisfies the strongest message hiding property, IND-CCA2, or does not satisfy even the weakest message hiding property, onewayness. We also showed that the computational PA2 notion is strictly stronger than the statistical one.

Keywords: Plaintext Awareness, Standard Model.

1 Introduction

The *Plaintext Awareness* [BR94, BDPR98, HLM03, BP04], which is also known as *PA2*, is a notion about the security of a public-key encryption scheme. Intuitively, we say that a public-key encryption scheme satisfies the PA2, if no adversary can generate a ciphertext “without knowing” the corresponding plaintext.

The PA2 notion is important, because it implies the chosen ciphertext security [BR94, BDPR98, BP04], if a public-key encryption scheme is the IND-CPA secure. Moreover, it is useful when one instantiates the ideal functions in the Dolev-Yao model [DY83], since the relation between the PA2 and the Dolev-Yao model is known [HLM03].

The original definition of the PA2 security was formalized in the random oracle model [BR94, BDPR98] and was highly dependent on this model, although the intuitive definition, mentioned above, does not depend on this model. Therefore, in the earlier study of the PA2, one of the main concerns was how to define the PA2 in the standard model.

In Asiacrypt 2004, Bellare and Palacio [BP04] succeeded in defining the standard model PA2. Their result is important, because we can analyze encryption schemes from the new view point whether these are PA2 secure. Here we briefly review their definition. They define PA2 notion based on the indistinguishability of two worlds, “Dec world”, and “Ext world”. An adversary in the Dec world can access the decryption oracle and so on. In contrast, the adversary in the Ext world can access an extractor, which simulates the decryption oracle, and so on. The extractor has to simulate the decryption oracle by using only data “which the adversary knows”. They define the three types of the PA2, named *perfect/statistical/computational* PA2, depending on that the Dec world and the Ext world are perfectly/statistically/computationally indistinguishable for the adversary.

They also succeeded in proving the fundamental theorem, which state that all of these plaintext awareness notions, together with IND-CPA security, imply the chosen ciphertext security.

1.1 Our Contributions

In this paper, we study the relationship between the standard model PA2 and the property about message hiding, that is, IND-CPA. At first glance, it seems that these two are independent notions. Indeed, it is well known that the random oracle model PA2 property does not imply the IND-CPA property and vice versa.

We however show that all of the perfect, statistical, and computational PA2 security in the standard model imply the IND-CPA security if the encryption function is oneway. Recall that the fundamental theorem that “(perfect, statistical, or computational) PA2 + IND-CPA \Rightarrow IND-CCA2” holds. Therefore, our result combining with the fundamental theorem shows the stronger variant of the fundamental theorem, “(perfect, statistical, or computational) PA2 + Oneway \Rightarrow IND-CCA2”. This result shows the “all-or-nothing” aspect of the PA2. That is, the standard model PA2 secure public-key encryption scheme either satisfies the strongest message hiding property, IND-CCA2, or does not satisfy even weakest message hiding property, onewayness.

Our result has not only theoretical interest but also can be useful when one prove the IND-CCA2 securities of public-key encryption schemes. Recall that it is non trivial to show the IND-CPA securities of some schemes satisfying the random oracle PA2, such as schemes with OAEP+ [OP01], 3-round OAEP [PP04], or Kobara-Imai [KI01] padding. However, in the case for schemes satisfying the standard model PA2, we are not required to prove the IND-CPA securities, since our result assures it.

We also study the gap between the computational and statistical PA2 securities. That is, we show that the computational PA2 security is strictly stronger than the statistical one. It is interesting to compare our result with Fujisaki’s result [F06] about the random oracle PA2. In his paper, he defined a *plaintext simulatability* (PS) notion, which was a “computational variant” of the random oracle PA2, and showed that plaintext simulatability notion was strictly

stronger than the random oracle PA2. Therefore, our result can be recognized as the standard model variant of Fujisaki's result [F06]. By comparing his result with our result, we can say that statistical and computational standard model PA2 notions are related to the random oracle PA2 and the PS, respectively.

We stress that, although our result and Fujisaki's result themselves are similar, these are of different model with different proof. Indeed we cannot use his proof because it highly depends on the random oracle model. Our proof is simpler and more intuitive than his.

1.2 Previous Works

Before the random oracle PA2 was defined, a weaker variant of it, named the random oracle *PA1* [BR94], had been defined. The first schemes satisfying the random oracle PA1 and PA2 were proposed in the paper of Bellare-Rogaway [BR94] and Fujisaki-Okamoto [FO99] respectively. In these papers, the authors proposed conversions which transform a trapdoor oneway permutation and an IND-CPA secure public-key encryption scheme to PA1 and PA2 secure public-key encryption scheme respectively. These conversions are called the OAEP and the Fujisaki-Okamoto conversions respectively.

Shoup [S01] showed that the random oracle PA1 + IND-CPA does not imply the IND-CCA2 security, although previously it had been thought that it did. In his paper, he also gave a revised version of the OAEP conversion, named the OAEP+, which transforms a trapdoor oneway permutation to a PA2 secure public-key encryption scheme on the random oracle model. The OAEP and other conversions satisfying a similar property are also studied in [CHJPPT98, B01, FOPS01, M01, OP01, CJNP02, KI01, KO03].

As far as we know, the first attempt to define the plaintext awareness not in the random oracle model was made by Herzog, Liskov, and Micali [HLM03]. They defined the PA2 notion on the key registration model [HLM03] and constructed a public-key encryption scheme which satisfies their PA2.

Bellare and Palacio [BP04] define not only the standard model PA2 but also the standard model PA1. They also showed that the Damgård [D91] and the lite Cramer-Shoup [CS01] public-key encryption schemes satisfy the standard model PA1 under the Diffie-Hellman Knowledge assumption [D91, BP04] and the DDH assumption. Later, Dent [D06] showed that the Cramer-Shoup public-key encryption scheme [CS98, CS01] satisfies the standard model PA2 security under the same assumption.

1.3 Organization

The paper is organized as follows: In Section 2, we review the definition of the standard model PA2. In Section 3, we show that the statistical PA2 is strictly stronger than the computational one. In Section 4, we show the main theorem, which states that “(perfect, statistical, or computational) PA2 + Oneway \Rightarrow IND-CPA”. Finally, in Section 5, we give the conclusion of our paper.

2 Definition of Standard Model PA2

In this section, we review the definition of the standard model PA2 [BP04]. Before giving the formal definition of the standard model PA2, we give intuitive explanation about it. The definition of the standard model PA2 is based on the indistinguishability of two worlds, named *Dec world* and *Ext world*, and uses entities named *adversary* and *extractor*. In the Dec world, the adversary can access to the decryption oracle and the encryption oracle. In contrast, the adversary in the Ext world can access to the extractor and the encryption oracle. The extractor has to simulate the decryption oracle by using only data “which the adversary can see”, that is, the adversary’s description, its random tape, and the answers from the encryption oracle.

It is a characteristic feature for the definition that it has a mechanism to hide the encryption query of the adversary from the extractor. In order to hide the encryption query, the entity, named *plaintext creator*, is also introduced. It is an entity which makes encryption queries as the adversary’s proxy. The adversary, in both Dec and Ext worlds, does not make encryption queries directly but sends an order to the plaintext creator, in order to make it send a query to the encryption oracle.

The extractor is not allowed to watch the plaintext creator’s random tape, although it is allowed to watch the adversary’s one. Hence it cannot know what queries are made to the encryption oracle. We say that an encryption scheme satisfies the standard model PA2, if the Dec and Ext worlds are indistinguishable for the adversary from each other.

We now define the standard model PA2 formally:

Definition 1 (Standard Model PA2 [BP04]). Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme. Let \mathcal{A} , \mathcal{P} , \mathcal{K} be polytime machines, which are respectively called *adversary*, *plaintext creator*, and *extractor*. Let $\mathcal{A}(\text{pk}; R_{\mathcal{A}})$ denotes the execution of an algorithm \mathcal{A} on inputting pk with the random coin $R_{\mathcal{A}}$. For a security parameter $\kappa \in \mathbb{N}$, we define two experiments $\mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{P}}^{\text{PA2-Dec}}(\kappa)$ and $\mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{K}, \mathcal{P}}^{\text{PA2-Ext}}(\kappa)$, shown in Fig. 1. In these experiments, it is required that \mathcal{A} makes no query (dec, C) for which $C \in \text{CList}$.

We say that the public-key encryption scheme Π is *perfectly/statistically/computationally standard model PA2* secure if

$$\forall \mathcal{A} \exists \mathcal{K} \forall \mathcal{P} : \mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{P}}^{\text{PA2-Dec}}(\kappa) \text{ and } \mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{K}, \mathcal{P}}^{\text{PA2-Ext}}(\kappa) \text{ are} \\ \text{perfectly/statistically/computationally indistinguishable for } \kappa.$$

Since we only discuss about the standard model PA2, we simply say that Π is *perfectly/statistically/computationally PA2* secure if it is perfectly/statistically/computationally standard model PA2 secure.

Theorem 2 (Fundamental Theorem for Standard Model PA2 [BP04]). *Let Π be an IND-CPA secure public-key encryption scheme. If Π is (perfect, statistical, or computational) PA2 secure, then Π is IND-CCA2 secure.*

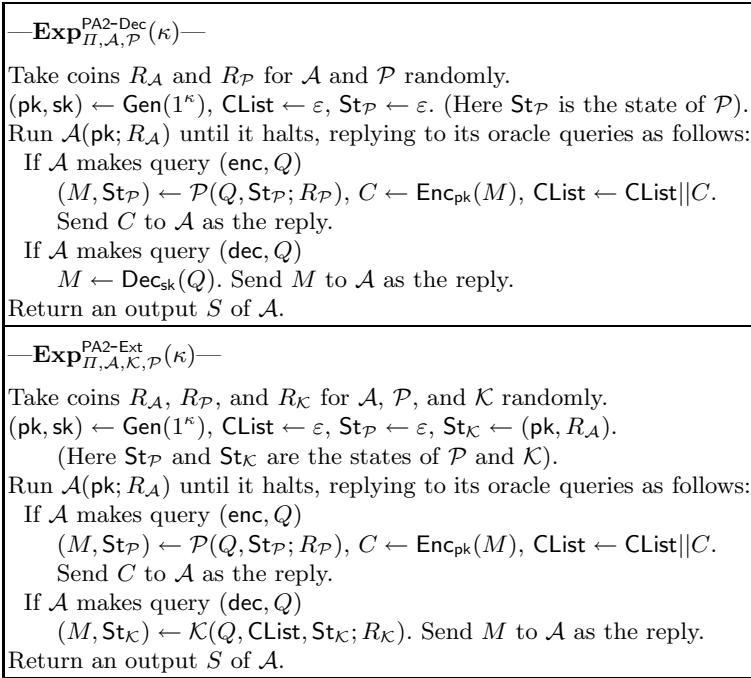


Fig. 1. Experiments used to define PA2 of [BP04]

3 Statistical PA2 Is Stronger Than Computational PA2

In this section, we show that the computational PA2 security is strictly stronger than the statistical one. That is, we give an example of a computational PA2 secure public-key encryption scheme $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ which is not statistical PA2 secure.

Let κ be a security parameter. Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme which is statistical PA2 secure and IND-CPA secure (and therefore IND-CCA2 secure). For instance, we can set Π to the Cramer-Shoup scheme [CS01], if the Diffie-Hellman Knowledge assumption [D91, BP04] and the DDH assumption holds. We construct the desired public-key encryption scheme $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ by modifying Π . The key generation algorithm $\text{Gen}'(1^\kappa)$ first executes $\text{Gen}(1^\kappa)$ and obtains a public key/secret key pair (pk, sk) as the output. After that, it selects a message M_0 randomly and computes a ciphertext $C_0 = \text{Enc}_{\text{pk}}(M_0)$. Then it sets $\text{pk}' = (\text{pk}, C_0)$ and $\text{sk}' = \text{sk}$. Finally, it outputs the public key/secret key pair (pk', sk') . We also set $\text{Enc}'_{\text{pk}'}(M) = \text{Enc}_{\text{pk}}(M)$ and $\text{Dec}'_{\text{sk}'}(C) = \text{Dec}_{\text{sk}}(C)$. See Fig. 2 also for the description of Π' .

We first see that Π' is not statistical PA2 secure. In order to see it, we construct an adversary \mathcal{A}'_0 such that no extractor can extract a message from the ciphertext output by \mathcal{A}'_0 . Our adversary \mathcal{A}'_0 is the one who obtains C_0 from its

$\text{Gen}'(1^\kappa)$: $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\kappa)$ Select a message M_0 randomly. $C_0 \leftarrow \text{Enc}_{\text{pk}}(M_0)$. $\text{pk}' \leftarrow (\text{pk}, C_0)$, $\text{sk}' \leftarrow \text{sk}$. Output (pk', sk') .
$\text{Enc}'_{\text{pk}'}(M) = \text{Enc}_{\text{pk}}(M)$, $\text{Dec}'_{\text{sk}'}(C) = \text{Dec}_{\text{sk}}(C)$.
$\mathcal{A}'_0(\text{pk}')$: Parse pk' as (pk, C_0) and output C_0 .

Fig. 2. Descriptions of $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ and \mathcal{A}'_0

input $\text{pk}' = (\text{pk}, C_0)$ and outputs C_0 . Recall that not \mathcal{A}'_0 but the key generation algorithm Gen' generates M_0 and C_0 . Therefore, \mathcal{A}'_0 “does not know” the message M_0 corresponding to C_0 . Since an extractor \mathcal{K}' is input only data which the adversary can see, \mathcal{K}' “cannot know” $M_0 = \text{Dec}'_{\text{sk}'}(C_0) = \text{Dec}_{\text{sk}}(C_0)$ either. This means that Π' is not statistical PA2 secure.

However, we can show that Π' is the computational PA2 secure. At first glance, it seems that Π' cannot be computational PA2 secure either, because even an extractor \mathcal{K}' for the computational PA2 “cannot know” $M_0 = \text{Dec}'_{\text{sk}'}(C_0)$ either. However, we actually do not require the extractor who “can know” such M_0 . Recall that the extractor \mathcal{K}' is only required to simulate the decryption oracle in such a way that an adversary \mathcal{A}'_0 cannot *computationally* distinguish the output of \mathcal{K}' from that of decryption oracle. Therefore, \mathcal{K}' does not need to output the plaintext M_0 itself, but can output the plaintext M_1 such that \mathcal{A}'_0 cannot computationally distinguish the distribution of M_1 from that of M_0 .

Recall that \mathcal{A}'_0 “knows” neither the plaintext M_0 nor the random number r which was used in the computation of $C_0 = \text{Enc}_{\text{pk}}(M_0; r)$. Recall also that Π satisfies the IND-CCA2 security. Hence, \mathcal{A}'_0 cannot distinguish a randomly selected message M_1 from M_0 . Therefore, \mathcal{K}' can output a randomly selected message M_1 as the answer to the decryption query C_0 .

Based on the above discussion, we can prove the following theorem.

Theorem 3. *Suppose that there exists at least one computational PA2 secure public-key encryption scheme. (For instance, if the Cramer-Shoup scheme [CS01] satisfies it under the DDH assumption and the Diffie-Hellman Knowledge assumption [D91, BP04]). Then there exists a computational PA2 secure public-key encryption which is not statistical PA2 secure.*

It is interesting to compare our result with Fujisaki’s result [F06] about the random oracle PA2. In his paper, he defined a *plaintext simulatability* (PS) notion, which was an “computational variant” of the random oracle PA2, and showed that plaintext simulatability notion was strictly stronger than the random oracle PA2. Therefore, our result can be recognized as the standard model variant of Fujisaki’s result [F06]. By comparing his result with our result, we can say

that statistical and computational standard model PA2 notions is related to the random oracle PA2 and the PS, respectively.

4 PA2-04 Together with Onewayness Implies IND-CPA

Our main result is the following:

Theorem 4. *Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme, which satisfies the onewayness property. If Π is perfectly, statistically, or computationally PA2 secure, then Π is IND-CPA secure, (and therefore IND-CCA2 secure).*

This result shows the “all-or-nothing” aspect of the PA2. That is, the (perfect, statistical, or computational) PA2 secure encryption scheme either satisfies the strongest message hiding property, IND-CCA2, or does not satisfy even the weakest message hiding property, onewayness.

Before proving Theorem 4, we see that one cannot remove the onewayness assumption from Theorem 4:

Theorem 5. *There is a public-key encryption which is perfect PA2 secure but is neither oneway nor IND-CPA secure.*

Proof (Theorem 5, sketch). Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme, such that an encryption $\text{Enc}_{\text{pk}}(M)$ of a message M is M itself. Then Π is clearly not IND-CPA secure. Recall the definition of the statistical PA2. We say that Π satisfies the statistical PA2 security if, for any adversary \mathcal{A} , there exists an extractor \mathcal{K} such that \mathcal{K} succeeds in extracting the plaintext M which corresponds to a ciphertext C output by \mathcal{A} . Since \mathcal{K} can know the message M directly from the ciphertext itself, Π satisfies the perfect PA2.

We first prove Theorem 4 for the special case where Π is statistically PA2 secure. Theorem 4 for the perfect PA2 security is clearly followed from it.

Proof (Theorem 4 for the statistical PA2, sketch). Let us make a contradictory supposition. That is, we suppose that there exists a statistically PA2 secure public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ which is not IND-CPA secure. Then we show that Π is not oneway.

In order to show it, we construct an adversary \mathcal{A}_0 which satisfies the following tricky property: \mathcal{A}_0 can obtain a ciphertext C_0 such that (1) \mathcal{A}_0 “does not know” the plaintext $M_0 = \text{Dec}_{\text{sk}}(C_0)$ and (2) C_0 is not generated by the encryption oracle. For a moment, suppose that we succeed in constructing such \mathcal{A}_0 . Since C_0 is not generated by the encryption oracle, \mathcal{A}_0 can make the query C_0 to the decryption oracle. Then, from the definition of the plaintext awareness, there exists an extractor \mathcal{K} which can extract the plaintext M_0 from the query C_0 of \mathcal{A}_0 . (Here we exploit the supposition that Π is statistically PA2 secure). This means that \mathcal{K} succeeds in outputting the unknown plaintext M_0 of a ciphertext C_0 . That is, \mathcal{K} can invert the encryption function Enc . This contradicts to the assumption that Π is oneway.

We next describe how to construct \mathcal{A}_0 . At first glance, it seems impossible to construct such \mathcal{A}_0 , since the definition of the plaintext awareness disable \mathcal{A}_0 generating a ciphertext C_0 “without knowing” the corresponding plaintext M_0 . The basic idea how \mathcal{A}_0 obtains such ciphertext C_0 is similar to that used in Section 3. In Section 3, the adversary obtains such C_0 from the key generation algorithm. In this proof, \mathcal{A}_0 obtains such C_0 from another entity, that is, a plaintext creator \mathcal{P}_0 . Then \mathcal{A}_0 “does not know” the message M_0 corresponding to C_0 , since not \mathcal{A}_0 itself but \mathcal{P}_0 generates C_0 . (We stress that not the encryption oracle but \mathcal{P}_0 itself generates C_0 . If the encryption oracle generates C_0 , \mathcal{A}_0 cannot send C_0 to the decryption oracle).

In order to employ the technique mentioned above, \mathcal{P}_0 has to send C_0 to \mathcal{A}_0 . However, there is no inherent communication channel which enables \mathcal{P}_0 to send C_0 directly to \mathcal{A}_0 . So, we construct a “virtual” communication channel from \mathcal{P}_0 to \mathcal{A}_0 .

Here we exploit the assumption that the public-key encryption scheme Π is not IND-CPA secure. Recall that the definition of the statistical PA2 security allows \mathcal{P}_0 to send plaintexts to the encryption oracle. Therefore, \mathcal{P}_0 can send to \mathcal{A}_0 a ciphertext c such that \mathcal{P}_0 generates the corresponding plaintext. Since Π is not IND-CPA secure, the ciphertext c leaks information of the corresponding plaintext. This means that \mathcal{P}_0 can send to \mathcal{A}_0 some sort of information via the ciphertext c . That is, \mathcal{P}_0 can use the ciphertext as the virtual channel.

We now describe more precisely how \mathcal{P}_0 “sends” C_0 to \mathcal{A}_0 . Let pk_0 be a public key and sk_0 be the unknown secret key corresponding to pk_0 . Since Π is not IND-CPA secure, there exist an algorithm \mathcal{B} , a state $\text{St}_{\mathcal{B}}$ of \mathcal{B} , a pair of messages (m_0, m_1) , and a non negligible and non negative valued function $\mu = \mu(\kappa)$ satisfying

$$\Pr(\mathcal{B}(\text{pk}_0, m_0, m_1, \text{Enc}_{\text{pk}_0}(m_1), \text{St}_{\mathcal{B}}) = 1) - \Pr(\mathcal{B}(\text{pk}_0, m_0, m_1, \text{Enc}_{\text{pk}_0}(m_0), \text{St}_{\mathcal{B}}) = 1) \geq \mu.$$

We set N to $\lceil 1/\mu \rceil$. Let b_i be the i -th bit of the ciphertext $C_0 = \text{Enc}_{\text{pk}_0}(M_0)$ such that M_0 is unknown. In advance, \mathcal{A}_0 sends $\text{pk}_0 || m_0 || m_1 || N$ to \mathcal{P}_0 , via the communication channel which enables \mathcal{A}_0 to query. For each i , \mathcal{P}_0 sends a message m_{b_i} as a query to the encryption oracle N times. Then the encryption oracle sends $c_1^{(i)} = \text{Enc}_{\text{pk}_0}(m_{b_i}), \dots, c_N^{(i)} = \text{Enc}_{\text{pk}_0}(m_{b_i})$ to \mathcal{A}_0 as the answers. After receiving $\{c_j^{(i)}\}$, \mathcal{A}_0 executes $\mathcal{B}(\text{pk}_0, m_0, m_1, c_j^{(i)}, \text{St}_{\mathcal{B}})$ and obtains an output $u_j^{(i)}$ of \mathcal{B} for each i and j . Then \mathcal{A}_0 sets $b'_i = 1$ if the number of j satisfying $u_j^{(i)} = 1$ is more than the number of j satisfying $u_j^{(i)} = 0$. Otherwise \mathcal{A}_0 sets $b'_i = 0$. Since \mathcal{B} has a non negligible advantage, the equality $u_j^{(i)} = b_i$ is satisfied with probability $1/2 + (\text{non negligible})$. Hence the equation $b'_i = b_i$ is satisfied with overwhelming probability. That is, \mathcal{A}_0 succeeds in reconstructing the bit b_i of the ciphertext C_0 for each i . Therefore, \mathcal{A}_0 can reconstruct the ciphertext $C_0 = b_1 || \dots || b_n$. In this way, \mathcal{A}_0 succeeds in “receiving” C_0 from \mathcal{P}_0 . \square

We now give the proof for the general case where Π satisfies only the computational PA2 security.

Proof (Theorem 4 for the computational PA2, sketch). As in the case of the proof of for statistical PA2, we suppose that there exists a computationally PA2 secure public-key encryption scheme Π which is not IND-CPA secure. Then we show that Π is not oneway.

We use similar algorithms to \mathcal{A}_0 and \mathcal{P}_0 of the proof for the statistical PA2. However, in the case of Π is computational PA2, the extractor \mathcal{K} may output a plaintext M' which is not equal to the plaintext $M_0 = \text{Dec}_{\text{sk}_0}(C_0)$, although the distribution of M' has to be computationally indistinguishable from that of M_0 . Therefore, in order to obtain M_0 , we modify the description of \mathcal{A}_0 and \mathcal{P}_0 .

We will first construct an adversary \mathcal{A}_1 by modifying \mathcal{A}_0 . Then, for some extractor \mathcal{K} , $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}'}^{\text{PA2-Ext}}(\kappa)$ is computationally indistinguishable from $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}'}^{\text{PA2-Dec}}(\kappa)$ for any \mathcal{P}' . Then, by modifying \mathcal{P}_0 , we will construct a plaintext creator \mathcal{P}_1 such that $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$ is, in fact, *statistically* indistinguishable from $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$, although we cannot exploit \mathcal{P}_1 itself to obtain the secret plaintext M_0 . We will finally construct a plaintext creator \mathcal{P}_2 , by modifying \mathcal{P}_1 , such that \mathcal{P}_2 can be exploited to obtain M_0 .

We will now give a brief description of \mathcal{A}_1 and \mathcal{P}_1 by describing the experiment $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$. (We stress that we first choose \mathcal{A}_1 , next obtain \mathcal{K} , and finally choose \mathcal{P}_1 , although we first describe about \mathcal{A}_1 and \mathcal{P}_1 , and next describe \mathcal{K} . One can easily check that we can take \mathcal{K} which does not depend on \mathcal{P}_1). In the experiment $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$, the experimenter first executes the key generation algorithm $\text{Gen}(1^\kappa)$ and obtains a public key/secret key pair (pk, sk) as an output. Then he inputs pk to the adversary \mathcal{A}_1 , the encryption oracle, and the decryption oracle. He also inputs sk to the decryption oracle. Then \mathcal{A}_1 executes $\mathcal{B}(\text{pk})$ and obtains $(m_0, m_1, \text{St}_{\mathcal{B}})$ as an output. After that, \mathcal{A}_1 sends $\text{pk}||m_0||m_1||N$ to \mathcal{P}_1 , via the communication channel which enables \mathcal{A}_1 to query. Here $N = \lceil 1/\mu \rceil$.

Then \mathcal{P}_1 generates a message M_1 randomly, and computes a ciphertext $C_1 = \text{Enc}_{\text{pk}}(M_1)$. After that, \mathcal{A}_1 and \mathcal{P}_1 execute the same procedures as those of \mathcal{A}_0 and \mathcal{P}_0 except that they execute these procedures using not C_0 but C_1 . That is, \mathcal{P}_1 “sends” C_1 to \mathcal{A}_1 via the “virtual” channel. After “receiving” C_1 from \mathcal{P}_1 , \mathcal{A}_1 makes query C_1 to the decryption oracle. Then the decryption oracle sends back a message M' to \mathcal{A}_1 as the answer to the query C_1 . (Note that the decryption oracle sends back a message $M' = M_1 = \text{Dec}_{\text{sk}}(C_1)$, although an extractor \mathcal{K} may send back a message M' other than M_1).

After that, \mathcal{A}_1 sends M' to \mathcal{P}_1 via the communication channel which enables \mathcal{A}_1 to query. \mathcal{P}_1 checks whether $M_1 = M'$ or not. Then \mathcal{P}_1 sets $S = 1$ if $M_1 = M'$, otherwise sets $S = 0$. After that, \mathcal{P}_1 “sends” S to \mathcal{A}_1 via the “virtual” channel. Finally, \mathcal{A}_1 outputs S .

Then, for some extractor \mathcal{K} , $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}'}^{\text{PA2-Ext}}(\kappa)$ is computationally indistinguishable from $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}'}^{\text{PA2-Dec}}(\kappa)$ for any \mathcal{P}' . In particular, $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$ is computationally indistinguishable from $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$.

We show that $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$ is, in fact, *statistically* indistinguishable from $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$. In the case where \mathcal{A}_1 and \mathcal{P}_1 are in the real experiment $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$, the output S of \mathcal{A}_1 is always 1. Recall that \mathcal{A}_1 cannot computationally distinguish $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$ from $\text{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$. Therefore, even in the experiment

$\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$, $S = 1$ is satisfied with overwhelming probability. Recall that $S = 1$ holds if and only if $M' = M$. Hence, \mathcal{K} succeeds in outputting the correct message M corresponding to $C'_1 = C_1 = \text{Enc}_{\text{pk}}(M)$ with overwhelming probability. This means that $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$ is statistically indistinguishable from $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{P}_1}^{\text{PA2-Dec}}(\kappa)$.

We next construct a plaintext creator \mathcal{P}_2 , by modifying \mathcal{P}_1 . Let (pk_0, C_0) be an instance of the onewayness game, and sk_0 be the unknown secret key corresponding to pk_0 . Our goal is to compute $M_0 = \text{Dec}_{\text{sk}_0}(C_0)$. The description of \mathcal{P}_2 is equal to that of \mathcal{P}_1 , except that (1) \mathcal{P}_2 takes C_0 as an input, (2) \mathcal{P}_2 does not use a ciphertext C_1 generated by \mathcal{P}_2 itself but instead uses a part C_0 of the instance (pk_0, C_0) of the onewayness game, and (3) \mathcal{P}_2 always sets $S = 1$.

We consider a modified version of the experiment $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_2}^{\text{PA2-Ext}}(\kappa)$, named $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_2}^{\text{PA2-Ext}^*}(\kappa, \text{pk}_0, C_0)$, in which the experimenter uses not the public key pk generated by $\text{Gen}(1^\kappa)$ but instead uses a part pk_0 of the instance (pk_0, C_0) of the onewayness game. Recall that both \mathcal{P}_1 in $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$ and \mathcal{P}_2 in $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_2}^{\text{PA2-Ext}^*}(\kappa, \text{pk}_0, C_0)$ set $S = 1$ with overwhelming probability. Moreover, the distribution of (pk_0, C_0) is equal to that of (pk, C) selected randomly. Hence, the behavior of \mathcal{P}_1 in $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$ is statistically indistinguishable from that of \mathcal{P}_2 in $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_2}^{\text{PA2-Ext}^*}(\kappa, \text{pk}_0, C_0)$. (Recall that \mathcal{K} is not input the random coin of a plaintext creator. Therefore, \mathcal{K} cannot distinguish the behavior of \mathcal{P}_1 from that of \mathcal{P}_2).

Therefore, the distribution of the output of $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_2}^{\text{PA2-Ext}^*}(\kappa, \text{pk}_0, C_0)$ is statistically indistinguishable from that of the output of $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$. Recall that, in the experiment $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_1}^{\text{PA2-Ext}}(\kappa)$, the output M' of \mathcal{K} is equal to $M_1 = \text{Dec}_{\text{sk}_0}(C_1)$ with overwhelming probability. Therefore, even in the experiment $\mathbf{Exp}_{\Pi, \mathcal{A}_1, \mathcal{K}, \mathcal{P}_2}^{\text{PA2-Ext}^*}(\kappa, \text{pk}_0, C_0)$, the output M' of \mathcal{K} is equal to $M_0 = \text{Dec}_{\text{sk}_0}(C_0)$ with overwhelming probability. This means that \mathcal{K} succeeds in obtaining the unknown plaintext $M_0 = \text{Dec}_{\text{sk}_0}(C_0)$ with overwhelming probability. \square

We see that Theorem 4 does not hold in the case of the random oracle PA2. See Appendix A for the definition of the random oracle PA2.¹

Proposition 6 *Suppose that there exists a group \mathcal{G} on which the DDH problem is easy although the CDH problem is hard. (For instance, we can set \mathcal{G} to an elliptic curve group on which a bilinear pairing [BF01, MOV93, JN03, SOK01] is defined). Then there exists a public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ which satisfies the random oracle PA2 security and the onewayness but does not satisfy the IND-CPA security.*

Proof (sketch). The desired encryption scheme is the Fujisaki-Okamoto [FO99] padded ElGamal encryption scheme such that a message and elements g and h of

¹ The definition of the random oracle PA2 differ subtly depending on papers. Our definitions are those of [BR94, FOPS01]. In some papers, such as [BDPR98, F06], the authors say that a public-key encryption scheme satisfies the random oracle PA2, if it satisfies both our definition and the IND-CPA security.

a public key (g, h) are taken from the above \mathcal{G} . Similar to the case of the original Fujisaki-Okamoto padded ElGamal encryption scheme, we can prove that the encryption scheme satisfies the random oracle model PA2 security. Moreover, it satisfies onewayness since the CDH problem is hard on \mathcal{G} . However, it does not satisfy the IND-CPA security since the DDH problem on \mathcal{G} is easy. \square

By applying the similar idea to the Damgård scheme [D91], one can also show that there exists a public-key encryption scheme which satisfies the standard model PA1 security [BP04] and the onewayness but does not satisfy the IND-CPA security. See Appendix A for the definition of the standard model PA1.

5 Conclusion

In this paper, we studied the relationship between the standard model PA2 and the property about message hiding, that is, IND-CPA. Although it seems that these two are independent notions at first glance, we showed that all of the perfect, statistical, and computational PA2 in the standard model imply the IND-CPA security if the encryption function is oneway. This result combining with the fundamental theorem implies the stronger variant of the fundamental theorem, “(perfect, statistical or computational) PA2 + Oneway \Rightarrow IND-CCA2”. It shows the “all-or-nothing” aspect of the PA2. That is, a (perfect, statistical, or computational) PA2 secure public-key encryption scheme either satisfies the strongest message hiding property, IND-CCA2, or does not satisfy even the weakest message hiding property, onewayness.

We also showed that the computational PA2 notion is strictly stronger than the statistical one. By comparing Fujisaki’s result [F06] with our result, we can say that statistical and computational standard model PA2 notions is related to the random oracle PA2 and the plaintext simulatability [F06], respectively.

Acknowledgements

We thank anonymous reviewers for helpful comments.

References

- [BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. CRYPTO 1998, pp.26-45.
- [BP04] Mihir Bellare, Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. ASIACRYPT 2004, pp. 48-62.
- [BR94] Mihir Bellare, Phillip Rogaway. Optimal Asymmetric Encryption. EUROCRYPT 1994, pp.92-111.
- [BR96] Mihir Bellare, Phillip Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. EUROCRYPT 1996, pp.399-416.
- [B01] Dan Boneh. Simplified OAEP for the RSA and Rabin Functions. CRYPTO 2001, pp.275-291.

- [BF01] Dan Boneh, Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. CRYPTO 2001, pp.213-229.
- [CHJPPT98] Jean-Sebastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, Christophe Tymen. Optimal Chosen-Ciphertext Secure Encryption of Arbitrary-Length Messages. PKC 2002, pp. 17-33.
- [CJNP02] Jean-Sebastien Coron, Marc Joye, David Naccache, Pascal Paillier. Universal Padding Schemes for RSA. CRYPTO 2002, pp. 226-241.
- [CS98] Ronald Cramer, Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. CRYPTO 1998, pp.13-25.
- [CS01] Ronald Cramer, Victor Shoup. Design and Analysis of Practical Public-Key Encryption Schemes. manuscript, 2001. Full version: SIAM J. Comp. 2004, 33(1), pp.167-226.
- [D91] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In CRYPTO'91, pp.445-456.
- [D06] Alexander W. Dent. Cramer-Shoup is Plaintext-Aware in the Standard Model. EUROCRYPT 2006.
- [DDN00] Danny Dolev, Cynthia Dwork, Moni Naor. Nonmalleable Cryptography. SIAM J. Comp. 2000, 30(2), pp. 391-437.
- [DY83] Danny Dolev, Andrew Chi-Chih Yao. On the security of public key protocols. IEEE Transactions on Information Theory, 1983, 29(2) pp.198-207.
- [F06] Eiichiro Fujisaki. Plaintext Simulatability. IEICE Trans. Fundamentals 2006, E89-A, pp.55-65, doi:10.1093/ietfec/e89-a.1.55. Preliminary version is available at <http://eprint.iacr.org/2004/218.pdf>
- [FO99] Eiichiro Fujisaki, Tatsuaki Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. PKC'99, pp. 53-68.
- [FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, Jacques Stern. RSA-OAEP Is Secure under the RSA Assumption. CRYPTO 2001, pp.260-274. J. Cryptology 2004, 17(2), pp.81-104.
- [HLM03] Jonathan Herzog, Moses Liskov, Silvio Micali. Plaintext Awareness via Key Registration. CRYPTO 2003, pp.548-564
- [JN03] Antoine Joux, Kim Nguyen. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups. J. Cryptology, 2003,16(4), pp.239-247. <http://eprint.iacr.org/2001/003>
- [KI01] Kazukuni Kobara, Hideki Imai. Semantically Secure McEliece Public-Key Cryptosystems-Conversions for McEliece PKC. In PKC 2001, pp. 19-35.
- [KO03] Yuichi Komano, Kazuo Ohta. Efficient Universal Padding Techniques for Multiplicative Trapdoor One-Way Permutation. CRYPTO 2003, pp. 366-382.
- [M01] James Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0. CRYPTO 2001, pp.230-238.
- [MOV93] Alfred Menezes, Tatsuaki Okamoto, Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. on Information Theory 1993, 39(5), pp.1639-1646.
- [OP01] Tatsuaki Okamoto, David Pointcheval. REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. CT-RSA 2001, pp.159-175.

- [PP04] Duong Hieu Phan, David Pointcheval. OAEP 3-Round: A Generic and Secure Asymmetric Encryption Padding. In *Asiacrypt 2004*. pp. 63-77.
- [SOK01] Ryuichi Sakai, Kiyoshi Ohgishi, Masao Kasahara. *Cryptosystems Based on Pairings*. SCIS 2001.
- [S00] Victor Shoup. Using Hash Functions as a Hedge against Chosen Ciphertext Attack. *EUROCRYPT 2000*, pp.275-288.
- [S01] Victor Shoup. OAEP Reconsidered. *CRYPTO 2001*, pp.239-259. *J. Cryptology*, 2002, 15(4), pp. 223-249.

A Definitions

A.1 Security Definitions of an Encryption Scheme

Definition 7 (IND-CPA/CCA1/CCA2). Let $\Pi=(\text{Gen},\text{Enc},\text{Dec})$ be a public-key encryption scheme and κ be a security parameter. For a public key/secret key pair (pk,sk) for Π , we let $\mathcal{O}_{\text{dec}}(\text{sk}, \cdot)$ be the oracle (named *decryption oracle*) such that it returns $\text{Dec}_{\text{sk}}(C)$ to an adversary when the adversary sends a ciphertext C to it. Let b be a bit. We also let $\mathcal{O}_{\text{enc}}(b, \text{pk}, \cdot)$ be the oracle (named *encryption oracle*) such that it returns $\text{Enc}_{\text{pk}}(M_b)$ to an adversary when the adversary sends a pair (M_0, M_1) of messages with the same length to it. We call $\text{Enc}_{\text{pk}}(M_b)$ the *challenge ciphertext*.

For a bit b and a polytime adversary \mathcal{A} , we set

$$\mathbf{P}_{\Pi, \mathcal{A}}^{(b)}(\kappa) = \Pr((\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\kappa), b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{enc}}(b, \text{pk}, \cdot), \mathcal{O}_{\text{dec}}(\text{sk}, \cdot)}(\text{pk}) : b' = 1),$$

and $\mathbf{Adv}_{\Pi, \mathcal{A}}^{\text{IND}}(\kappa) = |\mathbf{P}_{\Pi, \mathcal{A}}^{(1)}(\kappa) - \mathbf{P}_{\Pi, \mathcal{A}}^{(0)}(\kappa)|$.

Above, \mathcal{A} can make a query to $\mathcal{O}_{\text{enc}}(b, \text{pk}, \cdot)$ only once. Moreover, \mathcal{A} is not allowed to send the challenge ciphertext to $\mathcal{O}_{\text{dec}}(\text{sk}, \cdot)$.

We say that Π is *IND-CPA secure* if $\mathbf{Adv}_{\Pi, \mathcal{A}}^{\text{IND}}(\kappa)$ is negligible for any polytime adversary \mathcal{A} such that \mathcal{A} has made no query to $\mathcal{O}_{\text{dec}}(\text{sk}, \cdot)$. We say that Π is *IND-CCA1 secure* if $\mathbf{Adv}_{\Pi, \mathcal{A}}^{\text{IND}}(\kappa)$ is negligible for any polytime adversary \mathcal{A} such that \mathcal{A} has made no query to $\mathcal{O}_{\text{dec}}(\text{sk}, \cdot)$ after receiving the challenge ciphertext from $\mathcal{O}_{\text{enc}}(b, \text{pk}, \cdot)$. We also say that Π is *IND-CCA2 secure* if $\mathbf{Adv}_{\Pi, \mathcal{A}}^{\text{IND}}(\kappa)$ is negligible for any polytime adversary \mathcal{A} .

Definition 8 (Onewayness). Let κ be a security parameter, $\Pi=(\text{Gen},\text{Enc},\text{Dec})$ be a public-key encryption scheme, and \mathcal{M}_{pk} be a message space of Π in the case where the public key is pk . We say that Π is *oneway* (against CPA attack) if for any polytime adversary \mathcal{I} (named *inverter*), the probability

$$\Pr((\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\kappa), M \leftarrow \mathcal{M}_{\text{pk}}, C \leftarrow \text{Enc}_{\text{pk}}(M), M' \leftarrow \mathcal{I}(\text{pk}, C) : M = M')$$

is negligible for κ .

$\text{Hash} \leftarrow (\text{Set of all hash functions}), (\text{pk}, \text{sk}) \leftarrow \text{Gen}^{\text{Hash}}(1^\kappa).$
 $C \leftarrow \mathcal{A}^{\text{Hash}, \text{Enc}_{\text{pk}}^{\text{Hash}}}(\text{pk}).$
 $\text{HList} \leftarrow (\text{The list of all pairs of hash queries of } \mathcal{A} \text{ and the corresponding answers}),$
 $\text{CList} \leftarrow (\text{The list of all answers of the oracle } \text{Enc}_{\text{pk}}^{\text{Hash}}).$
 $M \leftarrow \mathcal{K}(\text{pk}, C, \text{HList}, \text{CList}).$
 If $M = \text{Dec}_{\text{sk}}^{\text{Hash}}(C)$, return 1. Otherwise return 0.

Fig. 3. Experiment used to define the random oracle PA2

Plaintext Awareness defined in [BR94, BDPR98]. We review the definitions of the PA1 and the PA2 in the random oracle model, defined in [BR94, BDPR98].

Definition 9 (Random Oracle PA2). Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme which uses a hash function. For a hash function Hash , we let Gen^{Hash} , Enc^{Hash} , and Dec^{Hash} denote the key generation, encryption, and decryption algorithms instantiated by the hash function Hash . Let \mathcal{A} and \mathcal{K} be polytime machines, which are respectively called *adversary* and *extractor*. For a security parameter $\kappa \in \mathbb{N}$, let $\mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{K}}^{\text{PA2-RO}}(\kappa)$ denote the experiment described in Fig. 3.

In this experiment, C must not be an element of CList . We say the public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is *random oracle PA2 secure*, if there exists \mathcal{K} such that, for any \mathcal{A} , the success probability

$$\text{Succ}_{\Pi, \mathcal{A}, \mathcal{K}}^{\text{PA2-RO}}(\kappa) = \Pr(\mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{K}}^{\text{PA2-RO}}(\kappa) = 1)$$

is overwhelming for κ .

Definition 10 (Random Oracle PA1). We say that a public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ satisfies the *random oracle PA1*, if there exists an extractor \mathcal{K} such that, for any adversary \mathcal{A} which makes no query to the encryption oracle, the success probability $\text{Succ}_{\Pi, \mathcal{A}, \mathcal{K}}^{\text{PA2-RO}}(\kappa)$ is negligible for κ .

Theorem 11 (Fundamental Theorem for the random oracle PA [BR94, BDPR98]). *Let Π be an IND-CPA secure public-key encryption scheme in the random oracle model. If Π satisfies the random oracle PA1 or PA2 security, then Π is IND-CCA1 or IND-CCA2 secure respectively.*

Standard Model PA1. We next review the definition of the PA1 in the sense of [BP04]. We use two experiments for defining PA1. These experiments are almost the same as those for PA2, except that an adversary makes no query to the plaintext creator \mathcal{P} . Since the experiments do not depend on \mathcal{P} , we denote them by $\mathbf{Exp}_{\Pi, \mathcal{A}}^{\text{PA1-Dec}}(\kappa)$ and $\mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{K}}^{\text{PA1-Ext}}(\kappa)$.

Definition 12 (standard model PA1). We say that a public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is *perfect/statistical/computational PA1 secure* in

the sense of [BP04], or easily *perfect/statistical/computational PA1 secure*, if for each adversary \mathcal{A} such that it makes no query to the plaintext creator, there exists \mathcal{K} such that the two experiments $\mathbf{Exp}_{\Pi, \mathcal{A}}^{\text{PA1-Dec}}(\kappa)$ and $\mathbf{Exp}_{\Pi, \mathcal{A}, \mathcal{K}}^{\text{PA1-Ext}}(\kappa)$ are perfectly/statistically/computationally indistinguishable. We simply say that Π is *PA1 secure* in the sense of [BP04], (or *PA1 secure*) if Π is computationally PA1 secure.

Theorem 13 (Fundamental Theorem for Standard Model PA1 [BP04]). *Let Π be an IND-CPA secure public-key encryption scheme. If Π is (perfect, statistical, or computational) PA1 secure, then Π is IND-CCA1 secure.*

On the Equivalence of RSA and Factoring Regarding Generic Ring Algorithms

Gregor Leander and Andy Rupp

Horst-Görtz Institute for IT-Security,
Ruhr-University Bochum, Germany
leander@itsec.rub.de, arupp@crypto.rub.de

Abstract. To prove or disprove the computational equivalence of solving the RSA problem and factoring integers is a longstanding open problem in cryptography. This paper provides some evidence towards the validity of this equivalence. We show that any efficient generic ring algorithm which solves the (flexible) low-exponent RSA problem can be converted into an efficient factoring algorithm. Thus, the low-exponent RSA problem is intractable w.r.t. generic ring algorithms provided that factoring is hard.

Keywords: Computational Equivalence, RSA Problem, Factorization Problem, Generic Algorithms.

1 Introduction and Related Work

The security of the well-known RSA encryption and signature scheme [1] relies on the hardness of the so-called RSA or root extraction problem: Let $n = pq$ be the product of two large primes and let e be a positive integer s.t. $\gcd(e, \phi(n)) = 1$. Then given n , e and an element $x \in \mathbb{Z}_n$, the challenge is to find an element $y \in \mathbb{Z}_n$ s.t. $y^e = x$. The RSA problem is closely related to the problem of factoring integers, i.e., in the case of an RSA modulus, finding p and q given n . While it is well-known that the RSA problem can be reduced to the factorization problem it is a longstanding open problem whether the converse is true, i.e., if an algorithm for finding e -th roots can be utilized in order to factor n efficiently.

Theoretical results towards disproving resp. proving the existence of such a reduction from the factorization to the RSA problem have been provided by Boneh and Venkatesan [2] resp. Brown [3]. In both papers the low-exponent variant of the RSA problem (LE-RSA) is considered, where the public exponent e is restricted to be smaller than some fixed constant or a product of small factors. Moreover, the results given in these papers are limited to (slight extensions) of straight line programs (SLPs). These are non-probabilistic algorithms only allowed to perform a fixed sequence of addition, subtraction and multiplication steps on their inputs without branching or looping. Thus, the result of such a program can be represented by a fixed integer polynomial in its inputs.

Boneh and Venkatesan [2] show that any straight line program that efficiently factors n given access to an oracle solving the LE-RSA problem can be converted

into a real polynomial-time factoring algorithm. This means, there exists no straight line reduction from factoring to LE-RSA, unless factoring is easy. The authors also show that this holds for algebraic reductions, which are straight line reductions extended by basic branching steps based on equality tests.

Recently, Brown [3] shows that any straight line program solving the LE-RSA problem also reveals the factorization of the RSA modulus. In other words, the LE-RSA problem is intractable for SLPs provided that factoring is hard. More precisely, he proves that an efficient SLP for breaking LE-RSA can always be transformed into an efficient factoring algorithm. Moreover, Brown outlines (see Appendix F in [3]) how this result extends to a generalization of SLPs (called SLEEPS) which are additionally allowed to perform basic branching steps based on the equality of elements.

At first sight, Brown's result seems to be contradictory to [2], since an SLP for breaking LE-RSA aids in factoring the modulus. However, the factoring algorithms considered by Brown which make use of the LE-RSA SLP are no straight line programs and in addition the LE-RSA SLP is not simply used as a black-box as it is done in [2]. So both results do not contradict but are results in opposite directions.

Another important theoretical result about the hardness of the RSA problem is due to Damgård and Koprowski [4]. They studied the problem of root extraction in finite abelian groups of unknown order and prove that both the standard and the flexible RSA problem, where the parameter e is no fixed input but can be chosen freely, are intractable w.r.t. generic *group* algorithms.

The concept of generic group algorithms has been introduced by Nechaev [5] and Shoup [6]. Loosely speaking, generic algorithms are probabilistic algorithms that given a group G as black box, may only perform a set of basic operations on the elements of G such as applying the group law, inversion of group elements and equality testing. Since the group is treated as black-box, the algorithms cannot exploit any special properties of the representation of group elements.

It is important to note that the generic algorithms for solving the (flexible) RSA problem considered in [4] are restricted in the following respects: They can only exploit the group structure of the multiplicative group \mathbb{Z}_n^* and not the full ring structure of \mathbb{Z}_n which would be more natural in the case of the RSA problem. Moreover, the RSA modulus n is not given as input to them. Instead, the multiplicative group is chosen at random according to a publicly known probability distribution and the algorithms know that the group order lies in a certain interval. Damgård and Koprowski leave it as an open problem to consider the RSA problem in a more natural generic model not having the restrictions described above.

1.1 Our Contribution

In this paper we propose a solution to the open problem stated in [4] by considering the hardness of the flexible LE-RSA problem w.r.t. to generic *ring* algorithms. We consider the following model of a generic ring algorithm: Let $\sigma : \mathbb{Z}_n \rightarrow S_n$, where $S_n \subset \{0, 1\}^{\lceil \log_2(n) \rceil}$ and $|S_n| = n$, denote a random encoding function for

\mathbb{Z}_n which is a function randomly chosen from the set of bijective mappings from \mathbb{Z}_n into the set of bit strings of sufficient length. A generic ring algorithm for the flexible RSA problem is a probabilistic algorithm which is given n , S_n and the encodings $\sigma(0)$, $\sigma(1)$ and $\sigma(x)$ as input. These encodings are the initial content of the *encoding list* which contains all encodings $\sigma(x_i)$ of ring elements x_i occurring during a computation. In a computation the algorithm can query a *ring oracle*, which given two indices i and j into this list computes $\sigma(x_i \pm x_j)$ or $\sigma(x_i x_j)$ and appends this encoding to the list. After some queries the algorithm finally outputs a pair $(e, \sigma(y))$ where $e > 1$ and $\gcd(e, \phi(n)) = 1$. It succeeds iff $y^e = x$.

Note that given the factorization of n , computing e -th roots is possible using $O(\log(n))$ oracle queries. So clearly it is not possible to prove that a generic ring algorithm given n needs exponential many oracle queries to solve the problem, since the algorithm might first factor n (without using the oracle) and then compute the e -th root using $O(\log(n))$ queries. Therefore any approach to prove something about the hardness of the problem in this model has to relate the RSA problem to the factorization problem.

We show that any efficient generic ring algorithm which solves the flexible LE-RSA problem with non-negligible probability can be converted into an efficient factoring algorithm having non-negligible probability. The considered generic algorithms can thereby only choose e from the set of exponents having some small fixed constant factor. Thus, the LE-RSA problem is intractable w.r.t. generic ring algorithms unless factoring is easy.

The paper at hand extends the results by Brown to a broader and more natural class of algorithms: First, the class of generic ring algorithms is clearly larger than the class of SLPs. Moreover, each SLEEP can be implemented as generic ring algorithm. However, it is not known if every generic ring algorithm can be realized as a SLEEP. We note that for part of our proof we use a Theorem given in [3].

2 Relating Flexible LE-RSA to Factoring

2.1 Generic Ring Algorithms

We formalize the notion of a generic algorithm for the ring \mathbb{Z}_n based on Shoup's generic group model [6]. To this end, the group oracle just needs to be extended by a multiplication operation in order to make the full ring structure of \mathbb{Z}_n available. However, the ring oracle \mathcal{O} we present slightly differs from such an extended group oracle in the following sense: Instead of using the ring \mathbb{Z}_n for the internal representation of ring elements, these elements are represented by polynomials in the variable X over \mathbb{Z}_n which are evaluated with x each time the encoding of a newly computed element must be determined. It is easy to see that both versions of a generic ring oracle are actually equivalent. However, we believe that the presented version is a better starting point for doing and understanding proofs in the generic model.

The generic oracle \mathcal{O} is defined as follows:

Input: As input \mathcal{O} receives $x \in_U \mathbb{Z}_n$, the modulus n and a list $\{\sigma_1, \dots, \sigma_n\}$ of n pairwise distinct bit strings randomly chosen from S_n .

Internal State: As internal state \mathcal{O} maintains two lists L and E which always have the same length. For an index $j \in \{1, \dots, |L|\}$ let L_j denote the j -th element of L and E_j the j -th element of E . In the list L , polynomials $L_j \in \mathbb{Z}_n[X]$ are stored which represent computed ring elements $L_j(x)$. The list E contains the encodings E_j of the corresponding ring elements $L_j(x)$. Moreover, \mathcal{O} maintains a counter c which counts the number of different elements contained in E and the encoding function $\sigma : \mathbb{Z}_n \rightarrow S_n$ which will be gradually defined during computation by the assignments between computed ring elements and the bit strings $\sigma_1, \dots, \sigma_n$.

Encoding elements: Each time a polynomial P is appended to the list L (during the initialization or query-handling phase described below) it is checked whether the corresponding element $P(x)$ has already been computed. More precisely, \mathcal{O} checks if there exists any $j \in \{1, \dots, |L|\}$ s.t.

$$(P - L_j)(x) \equiv 0 \pmod n.$$

If this is not the case, \mathcal{O} increases the counter c and appends the random bit string $\sigma_c \in S_n \setminus E$ to E which is different from all encodings so far contained in E . Additionally, the partially defined encoding function is updated with the new assignment, i.e., $\sigma := \sigma \cup \{P(x) \mapsto \sigma_c\}$. If the equation holds for any j the bit string E_j is again appended to E .

A run of \mathcal{O} consists of three phases:

Initialization: In this phase all lists are first set to the empty list, c is set to zero and the encoding function σ is set to be undefined for all $x \in \mathbb{Z}_n$. After that, L is appended with the polynomials $0, 1$ and X , E is appended with the respective encodings and σ and c are updated accordingly.

Query-handling: In the query-handling phase \mathcal{O} handles at most m queries. A query is a triple (\circ, j_1, j_2) where $\circ \in \{+, -, \cdot\}$ identifies an operation and j_1, j_2 are indices identifying the list elements the operation should be applied to. A query (\circ, j_1, j_2) is handled by computing the polynomial $P := L_{j_1} \circ L_{j_2}$, appending P to L and the respective encoding to E and updating σ and c accordingly.

Finalization: After an algorithm \mathcal{A} has made at most m queries to \mathcal{O} , it signals \mathcal{O} to finalize the computation before it eventually does its final output. Upon receiving this signal, \mathcal{O} updates the encoding function σ by assigning (in some fixed order) the $n - c$ ring elements $x \in \mathbb{Z}_n \setminus \{P(x) | P \in L\}$ which have not already occurred during computation to the random bit strings $\sigma_{c+1}, \dots, \sigma_n$. After that, \mathcal{O} signals \mathcal{A} to output its solution (e, out) , where $out \in S_n$, $e > 1$ and $\text{gcd}(e, \phi(n)) = 1$. We define the following success event

S : \mathcal{A} outputs an encoding $out = \sigma(y)$ and an integer e such that $y^e \equiv x \pmod n$.

2.2 Main Theorem

Our result lower bounding the hardness of flexible RSA in terms of the hardness of factoring integers can be stated as follows:

Theorem 1. *Let \mathcal{O} be a generic ring oracle for the ring \mathbb{Z}_n of order $n = pq$ as defined above. Let \mathcal{A} be a generic algorithm that makes at most m oracle queries to \mathcal{O} and let $(e, \sigma(y)) \leftarrow \mathcal{A}^{\mathcal{O}}(n, S_n, \sigma(0), \sigma(1), \sigma(x))$, where $e > 1$ and $\gcd(e, \phi(n)) = 1$, be its final output. Then the probability that y is an e -th root of x is upper bounded by*

$$\Pr[y^e = x] \leq (4\phi(e') + 2)\gamma + \frac{1}{n - m - 3},$$

where e' is the smallest factor of e and γ is a lower bound on the probability that n can be factored using \mathcal{A} and $O((\phi(e')^2 + \log(n))m^2)$ additional operations in \mathbb{Z}_n .

Note that the above theorem gives an upper bound on the probability that \mathcal{A} finds an e -th root which depends on the particular exponent e chosen by \mathcal{A} . More precisely, it is dependent on the size of the factors of e . This in particular means that we do not obtain a useful lower bound for exponents e consisting of “large” factors only. “Large” in this context means that the factors cannot be bounded by a polynomial in the security parameter $\log(n)$. However, if we restrict the class of allowed exponents \mathcal{A} can choose from to “low exponents”, i.e., exponents having at least one factor which is smaller than some fixed constant C , we always obtain a useful bound.

Corollary 1 (Hardness of Flexible LE-RSA). *Let \mathcal{O} be a generic ring oracle for the ring \mathbb{Z}_n of order $n = pq$ as defined above and let C be an arbitrary constant. Let \mathcal{A} be a generic algorithm that makes at most m oracle queries to \mathcal{O} and let $(e, \sigma(y)) \leftarrow \mathcal{A}^{\mathcal{O}}(n, S_n, C, \sigma(0), \sigma(1), \sigma(x))$ be its final output, where $e > 1$ has a factor smaller than C and $\gcd(e, \phi(n)) = 1$. Then the probability that y is an e -th root of x is upper bounded by*

$$\Pr[y^e = x] \leq (4C + 2)\gamma + \frac{1}{n - m - 3},$$

where γ is a lower bound on the probability that n can be factored using \mathcal{A} and $O((C^2 + \log(n))m^2)$ additional operations in \mathbb{Z}_n .

Let us assume that the number of queries m is polynomial bounded. Then observe that the probability γ is negligible if factoring is assumed to be hard since γ is a lower bound on the probability of factoring n using a polynomial bounded number of operations in \mathbb{Z}_n . Thus, in this case also the upper bound on the probability $\Pr[y^e = x]$ given in the corollary is negligible because m and C are polynomial bounded and γ is negligible. Hence, if factoring is hard Corollary 1 implies that the standard and the flexible LE-RSA problem are intractable w.r.t. generic ring algorithms. On the other hand, if for some special n root extraction is easy for generic algorithms, which might be possible, we know from our corollary that n can easily be factored.

Remark 1. In [4] special care has to be taken of the distribution of the group orders. More precisely, the order of the multiplicative group has to be randomly chosen according to certain so-called “hard” distributions in order to derive the

desired exponential lower bounds on the running time of generic group algorithms. This was an extension of Shoup's original model for the purpose of handling groups of hidden order. From this perspective things are easier in our model. As the order n of the additive group of the ring is given we do not have to worry about any special properties of the distribution according to which the order of the multiplicative group is chosen.

3 Proof of the Main Theorem

3.1 Outline

As usually done in proofs within the scope of the generic (group) model, we replace the original oracle \mathcal{O} with an oracle \mathcal{O}_{sim} that simulates \mathcal{O} without using the knowledge of the secret x . Then we show that the behavior of \mathcal{O}_{sim} is perfectly indistinguishable from \mathcal{O} unless a certain simulation failure \mathcal{F} occurs. From this, it immediately follows that the success probability of \mathcal{A} when interacting with \mathcal{O} is upper bounded by the sum of failure probability and the success probability of \mathcal{A} when interacting with \mathcal{O}_{sim} . We upper bound these probabilities in terms of the probability γ from Theorem 1 and the number of oracle queries.

Remark 2. The main difficulty in proving Theorem 1 is to bound the probability of a simulation failure \mathcal{F} . Usually, \mathcal{O}_{sim} is defined in a way that a simulation failure occurs iff two distinct polynomials $L_i, L_j \in L$ become equal under evaluation with x and one can determine a useful (i.e., negligible) upper bound on the probability of \mathcal{F} in terms of the maximal degree of such a difference polynomial $L_i - L_j$. However, here we face the problem that by using repeated squaring, \mathcal{A} can generate polynomials in L with exponential high degrees. Thus, we cannot derive non-trivial bounds anymore using this well-known technique. Note that this difficulty is inherent to the ring structure and does usually not occur when we consider cryptographic problems over generic groups. We solve it by simulating \mathcal{O} in a new way and relating the probability of \mathcal{F} to the probability γ .

3.2 The Simulation Game

The simulation oracle \mathcal{O}_{sim} is defined exactly like \mathcal{O} except that it determines the encoding of elements differently in order to be independent of the secret x . To this end, each time a polynomial P is appended to the end of list L (during initialization or query-handling), \mathcal{O}_{sim} does the following: Let $L_i = P$ denote the last entry of the updated list. Then for each $j < i$ the oracle chooses a random element $x_j^{(i)} \in_U \mathbb{Z}_n$ and checks whether

$$(L_i - L_j)(x_j^{(i)}) \equiv 0 \pmod{n}.$$

If this equation holds for some j_1, \dots, j_k the encoding E_j , where $j = \min(j_1, \dots, j_k)$, is appended as the encoding of the newly computed element to the list E .¹

¹ Note that it is not important how j is determined from $\{j_1, \dots, j_k\}$. j can be chosen from this set in an arbitrary way.

If no j exists s.t. the equation holds, counter c is increased and the random bit string $\sigma_c \in S_n \setminus E$ which is different from all encodings already contained in E is appended to E . Moreover, σ is updated by the assignment $P(x) \mapsto \sigma_c$.

Note that due to the modifications to the computation of encodings, it is now possible that both an element $P(x)$ is assigned to two or more different encodings and more than one element is assigned to the same encoding. Thus, the number $r_1 := n - c$ of unused encodings remaining after the query-handling phase may be greater or smaller than the number $r_2 := n - |\{P(x) | P \in L\}|$ of elements not occurring during computation. In the finalization phase \mathcal{O}_{sim} therefore assigns only $\min(r_1, r_2)$ elements from $\mathbb{Z}_n \setminus \{P(x) | P \in L\}$ to the encodings $\sigma_{c+1}, \dots, \sigma_{c+\min(r_1, r_2)}$ (but using the same order as \mathcal{O}).

Let us consider the following events which can occur in an interaction with the simulation oracle:

\mathcal{F} : There exists $i > j \in \{1, \dots, |L|\}$ such that

$$(L_i - L_j)(x) \equiv 0 \pmod n \text{ and } (L_i - L_j)(x_j^{(i)}) \not\equiv 0 \pmod n$$

or

$$(L_i - L_j)(x) \not\equiv 0 \pmod n \text{ and } (L_i - L_j)(x_j^{(i)}) \equiv 0 \pmod n.$$

\mathcal{S}_{sim} : \mathcal{A} outputs (e, out) such that out is the encoding of a unique element y and $y^e = x$.

The event \mathcal{S}_{sim} is the success event in a simulation game. The event \mathcal{F} is called simulation failure. It is important to observe that the original game and the simulation game proceed identically unless \mathcal{F} occurs: Assume that \mathcal{O} and \mathcal{O}_{sim} receive the same arbitrary but fixed input. Then issuing the same sequence of queries to \mathcal{O} and \mathcal{O}_{sim} results in the same sequence of encodings contained in E , the same sequence of polynomials contained in L and the same bijective encoding function σ , provided that \mathcal{F} does not happen. Furthermore, consider an algorithm $\bar{\mathcal{A}}$ with an arbitrary but fixed input on its random tape. Since $\bar{\mathcal{A}}$ is deterministic, it issues the same sequence of queries in both interactions if it receives the same sequence of encodings from \mathcal{O} and \mathcal{O}_{sim} . So assuming that \mathcal{F} does not happen, $\bar{\mathcal{A}}$ outputs the same exponent and encoding in both interactions and wins the simulation game if and only if it wins the original game. Thus, we have the following relation between the considered events

$$\mathcal{S} \wedge \neg \mathcal{F} \Leftrightarrow \mathcal{S}_{sim} \wedge \neg \mathcal{F}.$$

Using this relation we immediately obtain the desired upper bound on the success probability $\Pr[\mathcal{S}]$ in the original game in terms of the failure probability $\Pr[\mathcal{F}]$ and the probability $\Pr[\mathcal{S}_{sim} \wedge \neg \mathcal{F}]$ that no failure occurs and the algorithm succeeds in the simulation game.

$$\begin{aligned} \Pr[\mathcal{S}] &= \Pr[\mathcal{S} \wedge \neg \mathcal{F}] + \Pr[\mathcal{S} \wedge \mathcal{F}] \\ &= \Pr[\mathcal{S}_{sim} \wedge \neg \mathcal{F}] + \Pr[\mathcal{S} \wedge \mathcal{F}] \\ &\leq \Pr[\mathcal{S}_{sim} \wedge \neg \mathcal{F}] + \Pr[\mathcal{F}] \end{aligned}$$

In the following we relate these probabilities to the probability γ .

3.3 Simulation Failure Probability

For arbitrary but fixed indices $i > j \in \{1, \dots, m + 3\}$ we consider the difference polynomial $\Delta := L_i - L_j$. Let

$$N(\Delta) := \{a \in \mathbb{Z}_n \mid \Delta(a) \equiv 0 \pmod n\}$$

denote the set of zeros of this polynomial. Using the Chinese Remainder Theorem we can split $N(\Delta)$ into two sets

$$N(\Delta) \cong N^p(\Delta) \times N^q(\Delta), \text{ where}$$

$$N^p(\Delta) = \{a \in \mathbb{Z}_p \mid \Delta(a) \equiv 0 \pmod p\} \text{ and } N^q(\Delta) = \{a \in \mathbb{Z}_q \mid \Delta(a) \equiv 0 \pmod q\}.$$

Let the value $|N^p(\Delta)|/p$ be denoted by μ_Δ and $|N^q(\Delta)|/q$ by ν_Δ . The probability that a randomly chosen element $a \in_U \mathbb{Z}_n$ is a zero of the polynomial Δ can then be written as

$$\Pr[\Delta(a) \equiv 0 \pmod n; a \in_U \mathbb{Z}_n] = \nu_\Delta \mu_\Delta.$$

Thus, the probability $\Pr[\mathcal{F}_\Delta]$ that for a fixed polynomial Δ a simulation failure occurs is given by

$$\begin{aligned} \Pr[\mathcal{F}_\Delta] &= \Pr[\Delta(x) \equiv 0 \pmod n; x \in_U \mathbb{Z}_n] (1 - \Pr[\Delta(x_j^{(i)}) \equiv 0 \pmod n; x_j^{(i)} \in_U \mathbb{Z}_n]) \\ &\quad + \Pr[\Delta(x_j^{(i)}) \equiv 0 \pmod n; x_j^{(i)} \in_U \mathbb{Z}_n] (1 - \Pr[\Delta(x) \equiv 0 \pmod n; x \in_U \mathbb{Z}_n]) \\ &= 2\nu_\Delta \mu_\Delta (1 - \nu_\Delta \mu_\Delta). \end{aligned}$$

Now, we relate the failure probability $\Pr[\mathcal{F}_\Delta]$ with the probability γ from Theorem 1. First observe that if we can find an element

$$a \in ((\mathbb{Z}_p \setminus N^p(\Delta)) \times N^q(\Delta)) \cup (N^p(\Delta) \times (\mathbb{Z}_q \setminus N^q(\Delta))),$$

the polynomial Δ gives us the factorization of n by computing $\gcd(\Delta(a), n)$. Thus, the probability γ_Δ that the factorization can be revealed in this way by choosing a random $a \in_U \mathbb{Z}_n$ is given by

$$\gamma_\Delta = \mu_\Delta (1 - \nu_\Delta) + (1 - \mu_\Delta) \nu_\Delta = \mu_\Delta + \nu_\Delta - 2\mu_\Delta \nu_\Delta.$$

The crucial observation is the following lemma.

Lemma 1. *For any polynomial $\Delta \in \mathbb{Z}_n[X]$ it holds that $\Pr[\mathcal{F}_\Delta] \leq 2\gamma_\Delta$.*

Proof. We can see that $2\gamma_\Delta - \Pr[\mathcal{F}_\Delta] \geq 0$ by considering the following function:

$$\begin{aligned} f : \mathbb{R} \times \mathbb{R} &\rightarrow \mathbb{R} \\ f(\mu, \nu) &= (\mu\nu)^2 - 3\mu\nu + \mu + \nu \end{aligned}$$

In order to prove the lemma, we have to show that this function does not reach any negative values in $[0, 1]$. The only critical point in the set $[0, 1] \times [0, 1]$ and therefor the only possible extremum is

$$(\mu_0, \nu_0) = \left(\frac{\sqrt{3} - 1}{2}, \frac{\sqrt{3} - 1}{2} \right)$$

and we have $f(\mu_0, \nu_0) > 0$. Furthermore for the boundaries of the set $[0, 1] \times [0, 1]$ we get

$$\begin{aligned} f(0, \nu) &= \nu \geq 0, \\ f(\mu, 0) &= \mu \geq 0, \\ f(1, \nu) &= (\nu - 1)^2 \geq 0, \\ f(\mu, 1) &= (\mu - 1)^2 \geq 0. \end{aligned}$$

Thus it follows that for all $(\mu, \nu) \in [0, 1] \times [0, 1]$ we have $f(\mu, \nu) \geq 0$. □

Now, given \mathcal{A} consider an algorithm that evaluates all possible difference polynomials Δ with a randomly chosen element $a \in_U \mathbb{Z}_n$ and computes for each integer $\Delta(a)$ the value $\gcd(\Delta(a), n)$. The probability that n can be factored in this way is given by

$$\sum_{1 \leq j < i \leq |L|: \Delta := L_i - L_j} \gamma_{\Delta}.$$

The evaluation of all polynomials Δ can be done using a total of $O(m^2)$ operations. Computing all greatest common divisors requires $O(\log(n)m^2)$ operations using the Euclidean algorithm. So the probability of this factoring algorithm can be upper bounded by γ (cf. Theorem 1).

Using Lemma 1 we obtain the following bound on the probability of a simulation failure

$$\begin{aligned} \Pr[\mathcal{F}] &\leq \sum_{1 \leq j < i \leq |L|: \Delta := L_i - L_j} \Pr[\mathcal{F}_{\Delta}] \\ &\leq \sum_{1 \leq j < i \leq |L|: \Delta := L_i - L_j} 2\gamma_{\Delta} \\ &\leq 2\gamma. \end{aligned}$$

3.4 Success Probability in the Simulation Game

Let us split up the success event \mathcal{S}_{sim} in two sub-events: We say that the generic algorithm wins if either it outputs a new encoding $out \notin E$ corresponding to a unique element y which is an e -th root of x or if a polynomial in the list yields an e -th root when evaluated with the element x . We denote these events by

- \mathcal{S}_{sim}^1 : \mathcal{A} outputs (e, out) s.t. $out \notin E$, out is the encoding of a unique element y and $y^e = x$.
- \mathcal{S}_{sim}^2 : There exists a polynomial $P \in L$ s.t. $P(x)^e = x$.

Note that \mathcal{S}_{sim}^2 is more than actually needed: Here we do not require that \mathcal{A} actually outputs an encoding corresponding to $P(x)$, the existence of such a polynomial P in L is sufficient. We therefore have

$$\mathcal{S}_{sim} \Rightarrow \mathcal{S}_{sim}^1 \vee \mathcal{S}_{sim}^2$$

and thus

$$\Pr[\mathcal{S}_{sim} \wedge \neg \mathcal{F}] \leq \Pr[\mathcal{S}_{sim}^1 \wedge \neg \mathcal{F}] + \Pr[\mathcal{S}_{sim}^2].$$

Probability of Event $\mathcal{S}_{sim}^1 \wedge \neg \mathcal{F}$. Assume that the event \mathcal{F} has not happened during computation and \mathcal{A} outputs a pair (e, out) s.t. $out \notin E$. Since no simulation failure has occurred, σ is a bijective mapping and in particular the encodings $\sigma_{c+1}, \dots, \sigma_n$ not used in the query-handling phase are uniquely associated with the $n - c$ elements in $\mathbb{Z}_n \setminus \{P(x) | P \in L\}$. So the encoding out corresponds to a randomly chosen element $y \in \mathbb{Z}_n \setminus \{P(x) | P \in L\}$. Thus, we have

$$\Pr[\mathcal{S}_{sim}^1 \wedge \neg \mathcal{F}] \leq \Pr[\mathcal{S}_{sim}^1 \mid \neg \mathcal{F} \wedge out \notin E] \leq \frac{1}{n - (m + 3)}.$$

Probability of Event \mathcal{S}_{sim}^2 . Here we use the following Lemma which corresponds to (a slight extension of) Theorem 6 in [3].

Lemma 2. *Let $n = pq$, p, q prime and $e \in \mathbb{N}$ with $\gcd(e, \phi(n)) = 1$. Let a polynomial $P \in \mathbb{Z}_n[X]$ be given that can be evaluated for any element $x \in \mathbb{Z}_n$ using at most m additions and multiplications in \mathbb{Z}_n . For random $x \in_U \mathbb{Z}_n$ let the probability $\Pr[P(x)^e = x]$ be denoted by ε_P . Then using this polynomial n can be factored with probability*

$$\gamma_P \geq \frac{(e' - 1)(N - 1)}{\phi(e')e'N} \varepsilon_P$$

with at most $O(3\phi(e')^2m)$ operations in \mathbb{Z}_n , where e' is the smallest factor of e and N is the base of the natural logarithm.

The main idea behind this result is to evaluate P over an appropriate extension of \mathbb{Z}_n , where the mapping $x \mapsto x^{e'}$ is not a bijection anymore. Then one can use the well-known techniques to factor n given two different e' -th roots of the same element.

We now apply this result in our setting. First, observe that clearly all polynomials $P \in L$ can be evaluated using at most m operations in \mathbb{Z}_n . Thus, we can apply Lemma 2 to each P , i.e., we consider an algorithm that applies the procedure outlined in the proof of Theorem 6 in [3] to every polynomial in L . The running time of this algorithm is $O(\phi(e')^2m^2)$. The probability that n can be factored this way is given by $\sum_{P \in L} \gamma_P$ and by the definition of γ (cf. Theorem 1) it follows that

$$\sum_{P \in L} \gamma_P \leq \gamma.$$

Furthermore, it is easy to see that

$$\varepsilon_P \leq \frac{\phi(e')e'N}{(e' - 1)(N - 1)} \gamma_P \leq 4\phi(e')\gamma_P.$$

So we can conclude that the probability of the event \mathcal{S}_{sim}^2 is bounded by

$$\begin{aligned} \Pr(\mathcal{S}_{sim}^2) &\leq \sum_{P \in L} \varepsilon_P \\ &\leq \sum_{P \in L} 4\phi(e')\gamma_P \leq 4\phi(e')\gamma. \end{aligned}$$

3.5 Putting Things Together

Using the bounds on the probabilities in the simulation game we can bound the success probability in the original game. For a generic algorithm \mathcal{A} which makes m queries to \mathcal{O} and outputs a pair $(e, \sigma(y))$ consider an algorithm which

- chooses an element $a \in_U \mathbb{Z}_n$,
- computes $\gcd((L_i - L_j)(a), n)$ for each $i > j \in \{1, \dots, m+3\}$ and
- applies the procedure given in the proof of Theorem 6 in [3] to each L_i .

The running time of this algorithm is $O((\phi(e')^2 + \log(n))m^2)$ and by definition of γ its probability to factor n is less than γ .

Hence, the probability that y is an e -th root of the randomly chosen element x is bounded by

$$\begin{aligned} \Pr[y^e = x] &\leq \Pr[\mathcal{S}_{sim} \wedge \neg \mathcal{F}] + \Pr[\mathcal{F}] \\ &\leq \Pr[\mathcal{S}_{sim}^1 \wedge \neg \mathcal{F}] + \Pr[\mathcal{S}_{sim}^2] + \Pr[\mathcal{F}] \\ &\leq \frac{1}{n - m - 3} + 4\phi(e')\gamma + 2\gamma \\ &= (4\phi(e') + 2)\gamma + \frac{1}{n - m - 3}. \end{aligned}$$

This completes the proof of Theorem 1 and Corollary 1.

Acknowledgments. We would like to thank Ivan Damgård, Daniel Brown as well as the anonymous reviewers for their valuable comments.

References

1. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2) (1978) 120–126
2. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: *Advances in Cryptology: Proceedings of EUROCRYPT 1998*. Volume 1403 of *Lecture Notes in Computer Science.*, Springer-Verlag (1998) 59–71
3. Brown, D.R.L.: Breaking RSA may be as difficult as factoring. *Cryptology ePrint Archive*, Report 2005/380 (2006) <http://eprint.iacr.org/>.
4. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: *Advances in Cryptology: Proceedings of EUROCRYPT 2002*. Volume 2332 of *Lecture Notes in Computer Science.*, Springer-Verlag (2002) 256–271
5. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes* **55**(2) (1994) 165–172
6. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: *Advances in Cryptology: Proceedings of EUROCRYPT 1997*. Volume 1233 of *Lecture Notes in Computer Science.*, Springer-Verlag (1997) 256–266

Trading One-Wayness Against Chosen-Ciphertext Security in Factoring-Based Encryption

Pascal Paillier¹ and Jorge L. Villar²

¹ Cryptography Group, Security Labs, Gemalto
pascal.paillier@gemalto.com

² Departament de Matemàtica Aplicada, Universitat Politècnica de Catalunya
jvillar@ma4.upc.edu

Abstract. We revisit a long-lived folklore impossibility result for factoring-based encryption and properly establish that reaching maximally secure one-wayness (i.e. equivalent to factoring) and resisting chosen-ciphertext attacks (CCA) are incompatible goals for single-key cryptosystems. We pinpoint two tradeoffs between security notions in the standard model that have always remained unnoticed in the Random Oracle (RO) model. These imply that simple RO-model schemes such as Rabin/RW-SAEP[+]/OAEP[+][+], EPOC-2, etc. admit *no* instantiation in the standard model which CCA security is equivalent to factoring via a key-preserving reduction. We extend this impossibility to *arbitrary* reductions assuming non-malleable key generation, a property capturing the intuition that factoring a modulus n should not be any easier when given a factoring oracle for moduli $n' \neq n$. The only known countermeasures against our impossibility results, besides malleable key generation, are the inclusion of an additional random string in the public key, or encryption twinning as in Naor-Yung or Dolev-Dwork-Naor constructions.

1 Introduction

The Paradox. When a proof is given that some cryptosystem is semantically secure under chosen ciphertext attack (IND-CCA) under some complexity assumption, one generally checks whether one-wayness can be guaranteed under a weaker assumption. In the case of simple cryptosystems based on factoring large integers however, an inevitable tradeoff seems to exist between one-wayness and chosen ciphertext security. This incompatibility, which was observed for factoring-based signature schemes as well [20,22,13], is folklore knowledge and dates back to the late eighties. Despite early reasonings and attempts (later shown to be wrong) by a number of authors to formally prove it, this so-called “paradox” [13, Section 4] has remained essentially unexplored in a formal manner and, surprisingly enough, overlooked by contributors.

It is well known that the one-wayness of Rabin encryption and variants thereof [22,4,8,5] is equivalent to factoring (FACT), meaning that any efficient algorithm inverting encryption provides an efficient way to factor the modulus. It turns

out that the same algorithm can be used to totally break the cryptosystem (*i.e.* factor the modulus) under a trivial chosen ciphertext attack. This kind of attack has never been reported for RSA. But the one-wayness of RSA has not been shown to be equivalent to FACT. In fact, there is a separation result by Boneh and Venkatesan [6] which roughly tells that if a reduction from FACT to low-exponent RSA existed, then an efficient factoring algorithm could be constructed. Simultaneously, RSA-based cryptosystems such as OAEP [3] seem to resist chosen-ciphertext attacks convincingly well in practice. This provides the intuition that some sort of incompatibility must exist between achieving one-wayness under the weakest possible assumption (factoring) and achieving chosen ciphertext security at all.

In an early attempt to capture this intuition, Williams [22] makes the following (over)statement¹: *if the one-wayness of a factoring-based cryptosystem \mathcal{E} is equivalent to factoring then \mathcal{E} can be totally broken under chosen-ciphertext attack*. A simple proof for this claim was later shown to be incorrect by Goldwasser, Micali and Rivest [13], and the first public-key encryption scheme fully IND-CCA-secure under the factoring assumption was then discovered by Dolev, Dwork and Naor a few years later [10]. However, the incompatibility seems to persist for factoring-based encryption for which the public key consists of a single modulus.

Our Contributions. Our goal in this paper is to revisit [20,22,13] completely and clarify the conditions for such security incompatibilities to exist. We find that when properly formulated, certain security reductions for one-wayness and chosen-ciphertext security are indeed incompatible when considering *single-key* factoring-based encryption *i.e.* where the public key is just made of one hard-to-factor modulus. We reformulate the paradox observed by Williams in terms of *key-preserving* black-box reductions *i.e.* reductions which always call the adversarial oracle with the public-key they were given as input. We strengthen the original observation to show that if one can provide a key-preserving reduction from factoring to the (chosen-plaintext) semantic security of \mathcal{E} , then \mathcal{E} cannot fulfil *plaintext-checking* security. Plaintext-checking attacks, introduced in [18], assume that the attacker is given oracle access to a distinguishing oracle that tells whether a given ciphertext encrypts a given plaintext. It follows from combining these results that a wide class of factoring-based cryptosystems admit *no* key-preserving black-box reduction from factoring to breaking the security notions IND-CCA, OW-CCA and IND-PCA in the standard model. This provides black-box separations with well-known security proofs standing in the RO model [2] such as the one of Rabin-SAEP [5]. We provide later an explanation as to why these incompatibilities are avoided in the case of Naor-Yung [17] and Dolev-Dwork-Naor [10] constructions where public keys are composed of two or more independent moduli, as well as in the RO model.

Finally, we define the notion of non-malleable key generators, which formally captures the property that the factorizations of two public moduli n, n' where $n \neq n'$ are somehow “computationally independent” from one another. Similar

¹ The paradox appearing in [20,22,13] is discussed in the context of factoring-based signatures. This is a straightforward reformulation for factoring-based encryption.

notions of non-malleability for discrete logarithms recently appeared in [14,16]. Using non-malleability, we extend the scope of the previous impossibility results to *arbitrary* black-box reductions. Our refined results state that simple and innocuous-looking RO-secure factoring-based encryption schemes (e.g. Rabin-SAEP), when combined with non-malleable key generation, black-box separate the RO model from the standard model in a very strong sense: IND-CCA security is equivalent to FACT in the RO model while *no instantiation* of these schemes preserves such equivalence in the standard model.

We note that all impossibility results stated in this paper are easily transposed (*mutatis mutandis*) to factoring-based signature schemes. We do not treat the case of signatures here due to lack of space.

Roadmap. The paper is structured as follows. Section 2 gives preliminary facts about black-box reductions, single-key factoring-based encryption schemes and related security notions. Section 3 formally establishes the tradeoff between one-wayness and chosen ciphertext security. We also put forward a second tradeoff between semantic security against passive adversaries and plaintext-checking security. In Section 4, we give a formal definition of non-malleable instance generators and provide extended impossibility results. Section 5 discusses possible countermeasures such as encryption twinning to overcome these tradeoffs. We finally conclude on directions for further research in Section 6.

2 Preliminaries

Instance Generators. We define FACT as the problem of computing the list of all prime factors $\text{factors}(n) = (p_1, \dots, p_t)$ of a randomly chosen positive integer n . In cryptographic applications, one generally focuses on a specifically chosen distribution of hard instances by defining an instance generator Gen . Given a security parameter k , $\text{Gen}(1^k)$ generates a hard-to-factor modulus n , as well as the side information $\text{factors}(n)$. A probabilistic algorithm \mathcal{A} is said to (ε, τ) -break FACT [Gen] when

$$\Pr [(n, \text{factors}(n)) \leftarrow \text{Gen}(1^k) : \mathcal{A}(n) = \text{factors}(n)] \geq \varepsilon ,$$

where the probability is taken over the random coins of \mathcal{A} and Gen and \mathcal{A} halts after τ steps. FACT [Gen] is commonly referred to as the “factoring problem” when Gen is specified implicitly. For readability reasons, we may equivalently write $(n, \text{factors}(n)) \leftarrow \text{Gen}(1^k)$ or $n \leftarrow \text{Gen}(1^k)$ to state that n is drawn according to the distribution induced by $\text{Gen}(1^k)$. We note \mathcal{PK}_k the range of n i.e. the set of integers n such that $\Pr [n \leftarrow \text{Gen}(1^k)] > 0$ and $\mathcal{SK}_k = \text{factors}(\mathcal{PK}_k)$. Finally $\mathcal{PK} = \cup_k \mathcal{PK}_k$ and $\mathcal{SK} = \cup_k \mathcal{SK}_k$. Here are some instance generators commonly used in factoring-based encryption:

Rabin-Williams. Given 1^k , select uniformly at random two $\lceil k/2 \rceil$ -bit primes p and q such that $p \equiv 3 \pmod{8}$ and $q \equiv 7 \pmod{8}$. Set $n = pq$ and output $(n, (p, q))$.

- OU.** Given 1^k , randomly select two $\lceil k/3 \rceil$ -bit primes p and q . Set $n = p^2q$ and output $(n, (p, q))$.
- RSA- e .** Given a small integer e and 1^k , randomly select two $\lceil k/2 \rceil$ -bit primes p and q such that $\gcd(p-1, e) = \gcd(q-1, e) = 1$. Set $n = pq$ and output $(n, (p, q))$.
- Sophie-Germain.** Given 1^k , randomly select two $(\lceil k/2 \rceil - 1)$ -bit primes p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are also primes. Set $n = pq$ and output $(n, (p, q))$.

Single-Key Factoring-Based Encryption. A single-key factoring-based encryption scheme \mathcal{E} with security parameter k can be described as the combination of an instance generator Gen with a family of trapdoor functions on Gen , namely a pair (Enc, Dec) such that for any $n \in \mathcal{PK}$, $\text{Enc}(n, \cdot, \cdot)$ and $\text{Dec}(\text{factors}(n), \cdot)$ are integer-valued functions

$$\text{Enc}(n, \cdot, \cdot) : \mathbb{M}_n \times \mathbb{R}_n \rightarrow \mathbb{C}_n, \quad \text{Dec}(\text{factors}(n), \cdot) : \mathbb{C}_n \rightarrow \mathbb{M}_n$$

where \mathbb{M}_n , \mathbb{R}_n and \mathbb{C}_n denote respectively the plaintext, random and ciphertext spaces². We impose that for any $n \in \mathcal{PK}$, $m \in \mathbb{M}_n$ and $r \in \mathbb{R}_n$, $\text{Dec}(\text{factors}(n), \text{Enc}(n, m, r)) = m$. When $\text{Enc}(n, \mathbb{M}_n, \mathbb{R}_n) \subsetneq \mathbb{C}_n$, some elements of \mathbb{C}_n are not proper ciphertexts. When $c \notin \text{Enc}(n, \mathbb{M}_n, \mathbb{R}_n)$, $\text{Dec}(\text{factors}(n), c)$ returns a failure symbol $\perp \in \mathbb{M}_n$. We impose that $\text{Enc}(n, \cdot, \cdot)$ and $\text{Dec}(n, \cdot, \cdot)$ be efficiently computable for any arguments *i.e.* can be evaluated in time at most $\text{poly}(k)$ for $n \in \mathcal{PK}_k$. We identify $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec})$ to the three following probabilistic procedures:

- \mathcal{E} .**keygen:** Run $\text{Gen}(1^k)$ to get $(n, \text{factors}(n))$. The secret key is $\text{factors}(n)$ while the public key is n .
- \mathcal{E} .**encrypt:** Given a public key n and a message $m \in \mathbb{M}_n$, select $r \leftarrow \mathbb{R}_n$ uniformly at random and compute $c = \text{Enc}(n, m, r)$. The output ciphertext is $c \in \mathbb{C}_n$.
- \mathcal{E} .**decrypt:** Given the secret key $\text{factors}(n)$ and a ciphertext $c \in \mathbb{C}_n$, output $m = \text{Dec}(\text{factors}(n), c)$.

Examples of single-key factoring-based cryptosystems as defined above are countless: RSA³ and its numerous variants OAEP [3], REACT-RSA [18], PKCS#1 v1.5 [21], Rabin and related systems (Rabin-Williams [22], Blum-Goldwasser [4], Chor-Goldreich [8], Rabin-SAEP [5]), Naccache-Stern, Okamoto-Uchiyama and the EPOC family [12,11], Paillier [19] and variants. Many elliptic-curve-based cryptosystems such as KMOV [15], Vanstone-Zuccherato or Demytko [9] also fall into this category. We refer the reader to the extensive literature on factoring and its applications to cryptography for more detail.

² \mathbb{R}_n is the empty set when encryption is deterministic.

³ If the public exponent e is fixed (as usually done in practice), RSA decryption can be performed given the factors of n only.

Black-Box Reductions. Black-box reductions constitute a natural tool to relate computational problems and capture the way most security proofs are constructed. Given two computational problems P_1 and P_2 , a black-box reduction from P_1 to P_2 is a probabilistic algorithm \mathcal{R} which solves P_1 with the help of an oracle solving instances of P_2 . \mathcal{R} interacts with the oracle strictly as defined by the specification of P_2 and in particular has no view on the internal tapes of the oracle. The (extra) time of \mathcal{R} is the number of elementary steps required by \mathcal{R} to complete given that oracle calls count for one step by convention. A black-box reduction is polynomial when it runs in polynomial extra time (in a security parameter). It is crucial to remind that \mathcal{R} can be polynomial even when no polynomial-time algorithm solving P_2 is known to exist. We denote by $P_1 \Leftarrow P_2$ the fact that P_1 is polynomially black-box reducible to P_2 . We write $P_1 \Leftarrow_{\mathcal{R}} P_2$ when \mathcal{R} is known to reduce P_1 to P_2 . Polynomial equivalence is denoted by $P_1 \equiv P_2$. $\text{Succ}(P, \tau)$ stands for the maximal success probability of probabilistic algorithms solving P in no more than τ elementary steps. Similarly, $\text{Succ}(P_1 \Leftarrow P_2, \tau, \varepsilon, \ell)$ stands for the maximal success probability of probabilistic algorithms solving P_1 in no more than τ elementary steps and at most ℓ calls to an oracle solving P_2 with probability ε . All the reductions considered in this paper are black-box.

Security Notions for Factoring-Based Encryption. Security notions for encryption schemes are obtained by combining an adversarial goal with an attack model. **(Goals)** We say that an encryption scheme is *unbreakable* (UBK) when one cannot extract the secret key matching a prescribed public key. The scheme is said to be *one-way* (OW) when no adversary can recover a plaintext given its encryption. *Indistinguishability* (IND, a.k.a. *semantic security*) relates to the hardness of deciding whether a given ciphertext encrypts a given plaintext. **(Attacks)** We consider three attack models in this paper. In a *chosen-plaintext attack* (CPA), the adversary is given nothing more than the public key as input. In a *plaintext-checking attack* (PCA), the adversary is given access to a plaintext-checking oracle that tells whether a given ciphertext encrypts a given plaintext [18]. In a *chosen-ciphertext attack* (CCA), the adversary has access to a decryption oracle. Oracle access in OW-CCA, IND-PCA and IND-CCA games is limited in the sense that the adversary is not allowed to call the oracle on the challenge ciphertext itself. These definitions are classical. We refer to [1,18] for more detail on security notions for encryption schemes.

For convenience, we denote security notions in a positive fashion e.g. OW-PCA $[\mathcal{E}]$ denotes the problem of breaking the one-wayness of \mathcal{E} under plaintext-checking attack. This convention allows one to easily describe hierarchies between security notions using reductions. When the focus is on an adaptive attack (i.e. either PCA or CCA), we denote by ℓ -GOAL-ATK $[\mathcal{E}]$ the problem of breaking GOAL in no more than ℓ calls to the resource defined by ATK. Thus, breaking ℓ -IND-CCA $[\mathcal{E}]$ authorizes at most ℓ calls to the decryption oracle to break IND. We recall that GOAL-CCA $[\mathcal{E}] \Leftarrow$ GOAL-PCA $[\mathcal{E}] \Leftarrow$ GOAL-CPA $[\mathcal{E}]$ for any factoring-based encryption scheme \mathcal{E} and adversarial goal GOAL $\in \{\text{UBK}, \text{OW}, \text{IND}\}$. We also

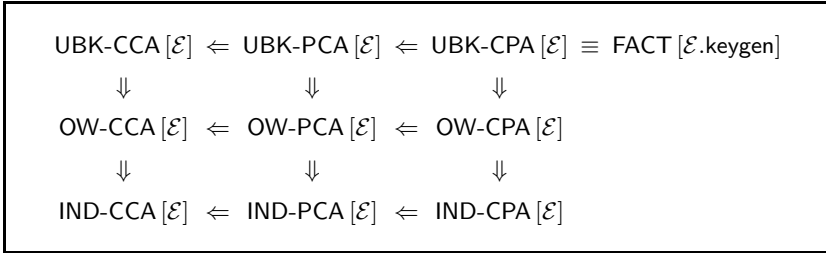


Fig. 1. Relations among security notions for single-key factoring-based encryption

have $\text{UBK-CPA}[\mathcal{E}] \equiv \text{FACT}[\mathcal{E}.\text{keygen}]$. We plot on Fig. 1 the map of security levels needed for the sake of this work.

3 Impossibility Results for Key-Preserving Reductions

In this section we focus on the standard-model security of single-key factoring-based encryption schemes. All black-box reductions known for such schemes are *key-preserving*, meaning informally that they make oracle calls to the adversary with the same key that they are given as input. We properly formalize this particular class of reductions in our setting⁴.

3.1 Key-Preserving Black-Box Reductions

Definition. We define key preservation for arbitrary security games related to a single-key factoring-based encryption scheme \mathcal{E} . Assume that $P_1[\mathcal{E}]$ and $P_2[\mathcal{E}]$ are two computational problems (view P_1 and P_2 as security notions) associated to \mathcal{E} . Consider a black-box reduction algorithm \mathcal{R} such that $P_1[\mathcal{E}] \Leftarrow_{\mathcal{R}} P_2[\mathcal{E}]$, meaning that \mathcal{R} makes oracle calls to an algorithm \mathcal{A} breaking $P_2[\mathcal{E}]$ to break $P_1[\mathcal{E}]$. Let $\text{Keys}(n, \text{aux}, \varpi)$ be the list (n_1, \dots, n_ℓ) of public keys given by \mathcal{R} as input to \mathcal{A} where (n, aux) is the modulus and auxiliary input for which \mathcal{R} has to break $P_1[\mathcal{E}]$ and $\varpi \in \{0, 1\}^{\text{poly}(k)}$ denotes the random tape of \mathcal{R} . Here the auxiliary input aux depends on the specification of P_1 . Note that the number ℓ of oracle calls is a deterministic function of n , aux and ϖ . \mathcal{R} is said to be key-preserving when for any aux, ϖ and $n \in \mathcal{PK}_k$, either $\ell = 0$ or $n_i = n$ for $i \in [1, \ell]$.

Key-preservation is transitive. It is obvious that if $P_1[\mathcal{E}] \Leftarrow_{\mathcal{R}_1} P_2[\mathcal{E}]$ and $P_2[\mathcal{E}] \Leftarrow_{\mathcal{R}_2} P_3[\mathcal{E}]$ such that \mathcal{R}_1 and \mathcal{R}_2 are both key-preserving, then there is a key-preserving reduction \mathcal{R}_3 such that $P_1[\mathcal{E}] \Leftarrow_{\mathcal{R}_3} P_3[\mathcal{E}]$.

Reductions among security notions are key-preserving. We use later the property that all the straightforward black-box reductions between the classical

⁴ A similar class of reductions for RSA encryption called *simple reductions* was recently considered by Brown [7].

security notions for \mathcal{E} such as $\text{IND-CCA}[\mathcal{E}] \Leftarrow \text{IND-PCA}[\mathcal{E}]$ and $\text{IND-CPA}[\mathcal{E}] \Leftarrow \text{OW-CPA}[\mathcal{E}]$ and so forth [1], are key-preserving.

3.2 One-Wayness Versus Chosen-Ciphertext Security

The following reformulates the observation made by Williams [22].

Theorem 1. *Let \mathcal{E} be a single-key factoring-based encryption scheme. If there exists a polynomial key-preserving black-box reduction \mathcal{R} such that $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow_{\mathcal{R}} \text{OW-CPA}[\mathcal{E}]$, then $\text{UBK-CCA}[\mathcal{E}]$ is polynomial.*

Proof. The main idea of the proof is basically a one-line statement and follows the reasoning of [22,13]. Let \mathcal{R} be such a key-preserving reduction algorithm, i.e. an algorithm that factors a modulus n randomly selected by $\mathcal{E}.\text{keygen}$ with non-negligible probability $\varepsilon_{\mathcal{R}}$ and extra time τ given black-box access to an adversary \mathcal{A} breaking $\text{OW-CPA}[\mathcal{E}]$ with probability at least ε . We construct an adversary \mathcal{M} against $\text{UBK-CCA}[\mathcal{E}]$.

Upon reception of the public key n in the UBK-CCA game, \mathcal{M} runs \mathcal{R} on input n and uses the decryption oracle to simulate the OW-CPA adversary. Since by definition the decryption oracle decrypts any ciphertext with probability $1 \geq \varepsilon$ in one elementary step, the simulation of \mathcal{A} is perfect for any $\varepsilon \in (0, 1)$. The simulation complies to the definition of \mathcal{R} because \mathcal{R} is key-preserving. It is therefore crucial that this property holds otherwise \mathcal{M} can by no means satisfy the queries \mathcal{R} makes to \mathcal{A} .

\mathcal{R} eventually returns the factorization of n with probability $\varepsilon_{\mathcal{R}}$ which \mathcal{M} then returns as output value. $\text{UBK-CCA}[\mathcal{E}]$ can therefore be broken with probability at least $\varepsilon_{\mathcal{R}}$ in extra time at most τ . □

3.3 Indistinguishability Versus Plaintext-Checking Security

Let us now consider $\text{IND-CPA}[\mathcal{E}]$. We know that there is a key-preserving reduction $\text{IND-CPA}[\mathcal{E}] \Leftarrow \text{OW-CPA}[\mathcal{E}]$ and also that key-preservation is transitive. Therefore Theorem 1 implies that there is no key-preserving reduction $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow \text{IND-CPA}[\mathcal{E}]$ unless $\text{UBK-CCA}[\mathcal{E}]$ is polynomial. But precisely because $\text{IND-CPA}[\mathcal{E}]$ is weaker than $\text{OW-CPA}[\mathcal{E}]$, a stronger incompatibility result can be found. We state:

Theorem 2. *Let \mathcal{E} be a single-key factoring-based encryption scheme. If there exists a polynomial key-preserving black-box reduction \mathcal{R} such that $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow_{\mathcal{R}} \text{IND-CPA}[\mathcal{E}]$, then $\text{UBK-PCA}[\mathcal{E}]$ is polynomial.*

Proof. Let us first describe in more detail the game played by a key-preserving reduction \mathcal{R} such that $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow_{\mathcal{R}} \text{IND-CPA}[\mathcal{E}]$. Given a modulus n , \mathcal{R} calls the adversarial oracle \mathcal{A} breaking $\text{IND-CPA}[\mathcal{E}]$ as follows. When \mathcal{R} calls $\mathcal{A}(\text{find}, n)$, \mathcal{A} outputs two plaintexts $m_0, m_1 \in M_n$ of equal length. \mathcal{R} then encrypts m_b for $b \leftarrow \{0, 1\}$ as c_b and calls $\mathcal{A}(\text{guess}, c_b)$. \mathcal{A} then returns its guess $\hat{b} \in \{0, 1\}$ to \mathcal{R} and $\Pr[\hat{b} = b] \geq \varepsilon$. We may assume w.l.o.g. that \mathcal{R} never calls

$\mathcal{A}(\text{guess}, c_b)$ before calling $\mathcal{A}(\text{find}, n)$ first and always calls $\mathcal{A}(\text{guess}, c_b)$ immediately after $\mathcal{A}(\text{find}, n)$, and that c_b is always a proper encryption of m_0 or m_1 . Let 2ℓ be the total number of calls to \mathcal{A} . Overall \mathcal{R} returns $\text{factors}(n)$ with probability $\varepsilon_{\mathcal{R}}$ and extra time τ .

We construct a trivial meta-reduction \mathcal{M} which converts the key-preserving black-box reduction \mathcal{R} into an adversary against UBK-PCA $[\mathcal{E}]$ and works with identical success probability in similar time. \mathcal{M} works as follows. Given a public key $n \leftarrow \mathcal{E}.\text{keygen}$, \mathcal{M} runs \mathcal{R} on input n and simulates the distinguisher \mathcal{A} using the plaintext-checking oracle of the UBK-PCA game. When \mathcal{R} calls $\mathcal{A}(\text{find}, n)$, \mathcal{M} returns two randomly selected plaintexts $m_0, m_1 \leftarrow \mathbf{M}_n$ of equal length. When \mathcal{R} calls $\mathcal{A}(\text{guess}, c_b)$, \mathcal{M} sends (m_1, c_b) to the plaintext-checking oracle and sends its output back to \mathcal{R} (recall that given $(m, c) \in \mathbf{M}_n \times \mathbf{C}_n$, the plaintext-checking oracle returns 1 if c encrypts m and 0 otherwise). Eventually \mathcal{R} stops and \mathcal{M} forwards the output of \mathcal{R} . By definition, the plaintext-checking oracle distinguishes plaintext-ciphertext pairs with probability one and \mathcal{M} therefore provides a perfect simulation of \mathcal{A} to \mathcal{R} for any $\varepsilon \in (0, 1)$. Hence \mathcal{M} outputs the factors of n with identical probability $\varepsilon_{\mathcal{R}}$ in time $\tau + 2\ell\rho(k)$ where $\rho(k) = \text{poly}(k)$ is the time needed to perform a random selection in \mathbf{M}_n . \square

3.4 Separation Results

Corollary 1. *Let \mathcal{E} be a single-key factoring-based encryption scheme. Unless $\text{FACT}[\mathcal{E}.\text{keygen}]$ is polynomial, there is no polynomial key-preserving black-box reduction $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow \text{IND-CCA}[\mathcal{E}]$.*

Proof. Assume that $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow_{\mathcal{R}_1} \text{IND-CCA}[\mathcal{E}]$ for some polynomial key-preserving black-box (PKPBB) reduction \mathcal{R}_1 . Since there exists a PKPBB reduction \mathcal{R}_2 such that $\text{IND-CCA}[\mathcal{E}] \Leftarrow_{\mathcal{R}_2} \text{OW-CPA}[\mathcal{E}]$, there must be a PKPBB reduction \mathcal{R}_3 such that $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow_{\mathcal{R}_3} \text{OW-CPA}[\mathcal{E}]$ by transitivity, resulting in that $\text{UBK-CCA}[\mathcal{E}]$ is polynomial by Theorem 1. Moreover since $\text{IND-CCA}[\mathcal{E}] \Leftarrow \text{UBK-CCA}[\mathcal{E}]$, one gets that $\text{IND-CCA}[\mathcal{E}]$ is polynomial and therefore that $\text{FACT}[\mathcal{E}.\text{keygen}]$ is polynomial as well. \square

Similar impossibility results are found for other security notions such as $\text{OW-CCA}[\mathcal{E}]$ and $\text{IND-PCA}[\mathcal{E}]$ using Theorem 2.

The Typical Example of Rabin-SAEP. We illustrate the importance of Corollary 1 by deducing a uninstantiability result for Rabin-SAEP. We first recall the definition of Rabin-SAEP [5]. Let s_m, s_0, s_1 be security parameters and $k = s_m + s_0 + s_1$. H denotes a fixed-size hash function $H : \{0, 1\}^{s_1} \rightarrow \{0, 1\}^{s_m + s_0}$. Here k plays the role of security parameter and the security proofs in [5] view s_m, s_0, s_1 as polynomial functions of k .

Rabin-SAEP.keygen : Given 1^k , generate a $(k + 2)$ -bit RSA modulus $n = pq$, $|p| = |q| = \lceil k/2 \rceil + 1$, $p = q = 3 \pmod{4}$ and $n \in [2^{k+1}, 2^{k+1} + 2^k)$. The secret key is $\text{factors}(n) = (p, q)$ while the public key is n .

Rabin-SAEP.encrypt : Given a public key n , the message space is $M_n = \{0, 1\}^{s_m}$ and the random space is $R_n = \{0, 1\}^{s_1}$. For $(m, r) \in M_n \times R_n$, $\text{Enc}(n, m, r)$ is defined as $((m \parallel 0^{s_0}) \oplus H(r)) \parallel r)^2 \bmod n$. The ciphertext space is $C_n = \mathbb{Z}_n$.

Rabin-SAEP.decrypt : Given $c \in C_n$ and (p, q) , compute $z_p = c^{(p+1)/4} \bmod p$ and $z_q = c^{(q+1)/4} \bmod q$. Output \perp if $z_p^2 \neq c \bmod p$ or $z_q^2 \neq c \bmod q$. Among the four values $\text{CRT}(\pm z_p, \pm z_q)$, select the only one y such that $y < n/2$ and y can be parsed as $((m \parallel 0^{s_0}) \oplus H(r)) \parallel r$ for some $(m, r) \in M_n \times R_n$. If this fails or can be done for more than one value for y , output \perp . Otherwise output m .

It is easily seen that Rabin-SAEP is a single-key factoring-based encryption scheme as per the definition of Section 2. We refer to [5, Section 4] for a proof that Rabin-SAEP is chosen-ciphertext secure under the factoring assumption in the RO model:

Theorem 3 (RO-model security of Rabin-SAEP [5]). *Let us view H as a random oracle. There exists a PKPBB reduction \mathcal{R} such that $\text{FACT}[\text{Rabin-SAEP.keygen}] \leftarrow_{\mathcal{R}} \text{IND-CCA}[\text{Rabin-SAEP}^H]$.*

We now state that for any instantiation of H , Rabin-SAEP does *not* admit a standard model counterpart of Theorem 3. This impossibility result comes as a direct application of Corollary 1.

Theorem 4 (Standard-model security of Rabin-SAEP). *Assuming $\text{FACT}[\text{Rabin-SAEP.keygen}]$ is intractable, there exists no PKPBB reduction $\text{FACT}[\text{Rabin-SAEP.keygen}] \leftarrow \text{IND-CCA}[\text{Rabin-SAEP}]$.*

Similar separations can be obtained for a wide range of factoring-based encryptions which chosen-ciphertext security is shown to be equivalent to factoring through key-preserving reductions in the RO model such as Rabin/RW-SAEP[+]/OAEP[+][+]/REACT, EPOC-2 [11], etc.

What Goes Wrong in the RO Model. Consider the meta-reduction \mathcal{M} in the proof of Theorem 1. \mathcal{M} cannot make any appropriate use of a key-preserving reduction \mathcal{R} standing in the RO model. In a typical random-oracle-based reduction, the random oracles of \mathcal{E} are simulated by \mathcal{R} . This additional power is beneficial to \mathcal{R} which introduces some form of correlation between its own variables and the responses of the simulated oracles. In a sense, \mathcal{R} is not totally black-box i.e. does not only rely on the input-output behavior of the OW-CPA adversary because \mathcal{R} controls the interactions between the adversary and the random oracles to increase its success probability.

In the chosen-ciphertext security game, however, the decryption oracle makes implicit calls (i.e. not controllable by any simulator) to the random oracles. Therefore, the meta-reduction cannot influence the decryption procedure by mimicking \mathcal{R} and consequently, can by no means correlate the internal variables of the decryption oracle to its own variables the same way \mathcal{R} does with the OW-CPA adversary. This explains why the RO model is unaware of incompatibilities in a general sense.

4 Extended Results for Non-malleable Key Generation

What we are after in this section is a way to strengthen the previous impossibility results. Recall we had to restrict the scope of Theorems 1 and 2 to key-preserving security reductions because the meta-reduction \mathcal{M} was unable to simulate the adversary \mathcal{A} when \mathcal{R} makes oracle calls to \mathcal{A} with arbitrary moduli. Our approach is to explicitly assume, as a property of the key generation of \mathcal{E} , that calling \mathcal{A} with $n' \neq n$ is essentially of no help to \mathcal{R} anyways. It appears that one faces definitional options when capturing this in a formal way: what we provide hereafter is the simplest definition that is strong enough for our purposes. This in turn allows us to consider *arbitrary* black-box reductions at the expense of making a complexity assumption on the key generation of \mathcal{E} .

4.1 Defining Non-malleable Generators

Intuition. An instance generator Gen is said to be malleable if factoring a randomly selected instance $n \leftarrow \text{Gen}(1^k)$ becomes substantially easier when given repeated access to an oracle which factors other instances $n' \neq n$ for $n' \in \mathcal{PK}_k$. A typical example of malleability is when \mathcal{PK}_k contains integers of variable size and number of prime factors. It is indeed trivial to factor n given an oracle that factors $n' = \alpha n$ if it happens that both n and n' are proper elements of \mathcal{PK}_k . We observe that most factoring-based cryptosystems define instance generators which precisely tend to avoid this malleability property by construction (see Section 2). What we need for our purposes is to define non-malleability in a strong sense.

Definition. To properly capture non-malleability, we define two games in which a probabilistic algorithm \mathcal{R} attempts to factor $n \leftarrow \text{Gen}(1^k)$ given access to an oracle $\mathcal{A}(n, \text{aux})$ solving with probability one some computational problem reducible to $\text{FACT}[\text{Gen}]$. Here, \mathcal{A} models the computational resources \mathcal{R} has access to and aux stands for any auxiliary input given to the oracle \mathcal{A} depending on how \mathcal{A} is specified. We may write $\mathcal{A}(n, \cdot)$ instead of $\mathcal{A}(n, \text{aux})$ to notify that aux is chosen freely and arbitrarily by \mathcal{R} when \mathcal{A} is called. Since we impose that oracle \mathcal{A} be perfect, we can abuse notations and identify \mathcal{A} to the problem solved by \mathcal{A} . A typical example of computational resources modelled by \mathcal{A} is when \mathcal{A} is polynomial (in which case \mathcal{R} is given no extra power), but one may consider problems reducible to $\text{FACT}[\text{Gen}]$ that do confer a computational advantage to \mathcal{R} , such as distinguishing quadratic residues modulo n , extracting e -th roots for $\text{gcd}(e, \phi(n)) = 1$ and so forth. In any case, we require \mathcal{A} to be perfectly reducible to $\text{FACT}[\text{Gen}]$ in polynomial time, that is, for any $n \in \mathcal{PK}_k$ and any admissible value for aux , $\mathcal{A}(n, \text{aux})$ must be solvable with probability one in time $t_{\mathcal{A}} = \text{poly}(k)$ given $\text{factors}(n)$. In **Game 0**, the success probability of \mathcal{R} is defined as

$$\text{Succ}_{\text{Gen}}^{\text{Game } 0}(\mathcal{R}, \mathcal{A}, \tau, \ell) = \Pr \left[n \leftarrow \text{Gen}(1^k) : \mathcal{R}^{\mathcal{A}(n, \cdot)}(n) = \text{factors}(n) \right]$$

where the probability is taken over the random tapes of \mathcal{R} and \mathcal{A} , \mathcal{R} runs in extra time at most τ and makes at most ℓ queries to $\mathcal{A}(n, \cdot)$. We further define

$$\text{Succ}_{\text{Gen}}^{\text{Game } 0}(\mathcal{A}, \tau, \ell) = \max_{\mathcal{R}} \text{Succ}_{\text{Gen}}^{\text{Game } 0}(\mathcal{R}, \mathcal{A}, \tau, \ell)$$

where the maximum is taken over all probabilistic algorithms \mathcal{R} playing Game 0. This can be interpreted as the success probability of the best reduction that makes use of $\mathcal{A}(n, \text{aux})$ to factor n for the given reduction parameters (τ, ℓ) . **In Game 1**, the reduction \mathcal{R} is given, in addition to \mathcal{A} , access to an auxiliary oracle $\text{FACT}(\cdot)$ that factors integers $n' \in \mathcal{PK}_k \setminus \{n\}$ with probability one. Its success probability $\text{Succ}_{\text{Gen}}^{\text{Game } 1}(\mathcal{R}, \mathcal{A}, \tau, \ell)$ is then

$$\Pr \left[n \leftarrow \text{Gen}(1^k) : \mathcal{R}^{\mathcal{A}(n, \cdot), \text{FACT}(\cdot)}(n) = \text{factors}(n) \right]$$

where the probability is taken over the random tapes of \mathcal{R} and \mathcal{A} , \mathcal{R} runs in extra time at most τ , makes $\ell_{\mathcal{A}}$ calls to $\mathcal{A}(n, \cdot)$ and ℓ_{FACT} calls of the type $\text{FACT}(n')$ with $n' \in \mathcal{PK}_k \setminus \{n\}$ such that $\ell_{\mathcal{A}} + \ell_{\text{FACT}} \leq \ell$. Let us define

$$\text{Succ}_{\text{Gen}}^{\text{Game } 1}(\mathcal{A}, \tau, \ell) = \max_{\mathcal{R}} \text{Succ}_{\text{Gen}}^{\text{Game } 1}(\mathcal{R}, \mathcal{A}, \tau, \ell)$$

where the maximum is taken over all probabilistic algorithms \mathcal{R} playing Game 1. This measures the success probability of the best reduction that uses simultaneously oracles $\mathcal{A}(n, \cdot)$ and $\text{FACT}(\cdot)$ to factor n in time τ and totalling no more than ℓ oracle calls. We finally define the *malleability* of Gen as

$$\Delta_{\text{Gen}}(\tau, \ell) = \max_{\mathcal{A} \Leftarrow \text{FACT}[\text{Gen}]} \left| \text{Succ}_{\text{Gen}}^{\text{Game } 1}(\mathcal{A}, \tau, \ell) - \text{Succ}_{\text{Gen}}^{\text{Game } 0}(\mathcal{A}, \tau, \ell) \right|,$$

where the maximum is now taken over all computational problems \mathcal{A} perfectly reducible to $\text{FACT}[\text{Gen}]$ in polynomial time.

Remark 1. It is easily seen that $\Delta_{\text{Gen}}(\tau, 0) = 0$ for any $\tau \geq 0$.

Definition 1 (Non-Malleable Instance Generators). *We say that an instance generator Gen is non-malleable when $\Delta_{\text{Gen}}(\tau, \ell)$ remains polynomially negligible in k when $\tau = \text{poly}(k)$ and $\ell = \text{poly}(k)$.*

Remark 2. The purpose of Game 0 is to include all key-preserving reductions \mathcal{R} such that $\text{FACT}[\text{Gen}] \Leftarrow_{\mathcal{R}} \mathcal{A}$. Since the success probability ε of the adversarial oracle plays no role in the proofs of Theorems 1 and 2, these can be reformulated as follows. For any positive integers τ, ℓ :

Th. 1: $\text{Succ}_{\mathcal{E}. \text{keygen}}^{\text{Game } 0}(\text{OW-CPA}[\mathcal{E}], \tau, \ell) \leq \text{Succ}(\ell\text{-UBK-CCA}[\mathcal{E}], \tau)$

Th. 2: $\text{Succ}_{\mathcal{E}. \text{keygen}}^{\text{Game } 0}(\text{IND-CPA}[\mathcal{E}], \tau, \ell) \leq \text{Succ}(\ell\text{-UBK-PCA}[\mathcal{E}], \tau + 2\ell\rho(k))$

4.2 A Fundamental Lemma

We now come back to our earlier discussion about extending the scope of Theorem 1 and dealing with \mathcal{R} calling \mathcal{A} with arbitrary moduli $n' \neq n$. The oracle calls \mathcal{R} makes to \mathcal{A} are now of two types: calls with the same modulus n (key-preserving calls) and calls with $n' \neq n$ (non-key-preserving calls). Our definition of non-malleability allows us to limit the computational advantage conferred to \mathcal{R} by its non-key-preserving calls.

Lemma 1. *Let Gen be an instance generator and let \mathcal{A} be a computational problem perfectly reducible to $\text{FACT}[\text{Gen}]$ in time $t_{\mathcal{A}}$. Then for any positive integers τ, ℓ and any $\varepsilon \in (0, 1)$,*

$$\text{Succ}(\text{FACT}[\text{Gen}] \Leftarrow \mathcal{A}, \tau, \varepsilon, \ell) \leq \text{Succ}_{\text{Gen}}^{\text{Game } 1}(\mathcal{A}, \tau + \ell \cdot t_{\mathcal{A}}, \ell).$$

Proof. Recall that \mathcal{A} denotes a computational problem here. Assume $\mathcal{R}(\tau, \varepsilon, \ell)$ solves $\text{FACT}[\text{Gen}] \Leftarrow \mathcal{A}$ i.e. factors $n \leftarrow \text{Gen}(1^k)$ in extra time τ with no more than ℓ calls to an oracle $\mathcal{A}_{\mathcal{R}}$ solving \mathcal{A} with probability ε . Let $\varepsilon_{\mathcal{R}}$ be the success probability of \mathcal{R} . We construct an algorithm \mathcal{M} which plays Game 1 with respect to a perfect oracle $\mathcal{A}_{\mathcal{M}}$ for \mathcal{A} and succeeds with identical probability and similar running time. Algorithm \mathcal{M} works as follows. Given a randomly selected modulus $n \leftarrow \text{Gen}(1^k)$, \mathcal{M} runs \mathcal{R} on input n . Now when \mathcal{R} calls $\mathcal{A}_{\mathcal{R}}(n, \text{aux})$, \mathcal{M} calls $\mathcal{A}_{\mathcal{M}}(n, \text{aux})$ and forwards the output to \mathcal{R} . When \mathcal{R} calls $\mathcal{A}_{\mathcal{R}}(n', \text{aux})$ for $n' \in \mathcal{PK}_k \setminus \{n\}$, \mathcal{M} calls $\text{FACT}(n')$ to get $\text{factors}(n')$ and solves $\mathcal{A}(n', \text{aux})$ in time $t_{\mathcal{A}}$. \mathcal{M} then returns the result to \mathcal{R} . \mathcal{R} eventually stops and \mathcal{M} returns the output of \mathcal{R} . The simulation of $\mathcal{A}_{\mathcal{R}}$ is perfect for any $\varepsilon \in (0, 1)$. \mathcal{M} requires extra time at most $\tau + \ell \cdot t_{\mathcal{A}}$ and makes at most ℓ calls to oracles $\mathcal{A}_{\mathcal{M}}$ and $\text{FACT}(\cdot)$ altogether. \square

4.3 Extended Separation Results

Theorem 5. *Let \mathcal{E} be a single-key factoring-based encryption scheme and assume $\mathcal{E}.\text{keygen}$ is non-malleable. If $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow \text{OW-CPA}[\mathcal{E}]$ then $\text{UBK-CCA}[\mathcal{E}]$ is polynomial.*

Proof. Let us consider $\mathcal{A} = \text{OW-CPA}[\mathcal{E}]$. Obviously \mathcal{A} is perfectly reducible to $\text{FACT}[\mathcal{E}.\text{keygen}]$ since given any $n \in \mathcal{PK}_k$, $\text{aux} = c \in \mathcal{C}_n$ and $\text{factors}(n)$, $\mathcal{A}(n, \text{aux})$ is solved by computing $m = \text{Dec}(\text{factors}(n), c)$ in time $t_{\mathcal{A}} = \text{poly}(k)$. Applying Lemma 1, we get for any τ, ℓ and $\varepsilon \in (0, 1)$:

$$\begin{aligned} & \text{Succ}(\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow \text{OW-CPA}[\mathcal{E}], \tau, \varepsilon, \ell) \\ & \leq \text{Succ}_{\mathcal{E}.\text{keygen}}^{\text{Game } 1}(\text{OW-CPA}[\mathcal{E}], \tau + \ell \cdot \text{poly}(k), \ell) \\ & \leq \text{Succ}_{\mathcal{E}.\text{keygen}}^{\text{Game } 0}(\text{OW-CPA}[\mathcal{E}], \tau + \ell \cdot \text{poly}(k), \ell) + \Delta_{\text{Gen}}(\tau + \ell \cdot \text{poly}(k), \ell) \\ & \leq \text{Succ}(\ell\text{-UBK-CCA}[\mathcal{E}], \tau + \ell \cdot \text{poly}(k)) + \Delta_{\text{Gen}}(\tau + \ell \cdot \text{poly}(k), \ell). \end{aligned}$$

We now extend asymptotically the above to $\tau, \ell = \text{poly}(k)$. Since $\mathcal{E}.\text{keygen}$ is non-malleable, the malleability term $\Delta_{\text{Gen}}(\tau + \ell \cdot \text{poly}(k), \ell)$ remains negligible.

Since $\text{Succ}(\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow \text{OW-CPA}[\mathcal{E}], \tau, \varepsilon, \ell)$ is non-negligible by assumption, $\text{Succ}(\ell\text{-UBK-CCA}[\mathcal{E}], \tau + \ell \cdot \text{poly}(k))$ must be non-negligible as well, thereby giving the result. \square

The same proof technique applies to $\text{IND-CPA}[\mathcal{E}]$ and shows that there exists no reduction $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow \text{IND-CPA}[\mathcal{E}]$ unless $\text{UBK-PCA}[\mathcal{E}]$ is polynomial or $\mathcal{E}.\text{keygen}$ is malleable. Based on a reasoning similar to the proof of Corollary 1, we deduce from these incompatibilities that:

Corollary 2. *Let \mathcal{E} be a single-key factoring-based encryption scheme and assume $\mathcal{E}.\text{keygen}$ is non-malleable. There is no polynomial black-box reduction $\text{FACT}[\mathcal{E}.\text{keygen}] \Leftarrow \text{IND-CCA}[\mathcal{E}]$ unless $\text{FACT}[\mathcal{E}.\text{keygen}]$ is polynomial.*

To exemplify Corollary 2, we provide this extended impossibility result for Rabin-SAEP.

Theorem 6 (Standard-model security of Rabin-SAEP, revisited). *Assume $\text{Rabin-SAEP}.\text{keygen}$ is non-malleable. Then Rabin-SAEP admits no instantiation in the standard model which is chosen-ciphertext secure under the factoring assumption i.e. for any instantiation of H ,*

$$\text{IND-CCA}[\text{Rabin-SAEP}] \not\equiv \text{FACT}[\text{Rabin-SAEP}.\text{keygen}] .$$

Similar uninstantiability results hold for single-key factoring-based encryption schemes which chosen-ciphertext security is shown to be equivalent to factoring in the RO model. Again, these stronger separations are effective only when the underlying key generation is non-malleable. In other words, either these encryption schemes do separate the RO model from the standard model in a very strong sense, or their key generation must be malleable along the lines of Definition 1.

5 Overcoming Uninstantiability

Keyed Paddings. At first look, including some additional key material such as a random string in the public key seems to invalidate our impossibility results completely. Typically the extra parameter can serve as a function index in a keyed family of hash functions. This seems to be an efficient countermeasure for single-key factoring-based encryption making use of encryption paddings which, unlike SAEP[+]/OAEP[+][+], Fujisaki-Okamoto and REACT, include keyed hash functions.

Encryption Twinning. Naor and Yung [17] and Dolev, Dwork and Naor [10] suggested transformations which when applied to IND-CPA -secure encryptions such as Blum-Goldwasser [4] or Chor-Goldreich [8] may lead to IND-CCA -secure schemes under the factoring assumption. The transformed schemes use public keys containing two or more independently generated moduli with respect to the basic scheme. This paradigm makes it possible to generically construct a larger class of factoring-based cryptosystems which IND-CCA -security can possibly be proven equivalent to factoring, thereby escaping all incompatibility results described earlier. We comment that the cornerstone of Theorem 1

resides in that the decryption oracle provided in the UBK-CCA game can serve as a factoring algorithm when interfaced with the black-box reduction \mathcal{R} . We now see how encryption twinning prohibits such a use of the decryption oracle. The public key in a Naor-Yung-transformed encryption scheme $\text{NY}(\mathcal{E})$ is (n_1, n_2, r) where $n_1, n_2 \leftarrow \mathcal{E}.\text{keygen}$ and r is a random string used to generate NIZK proofs during encryption. An encryption of $m \in \mathbb{M}_{n_1} \cap \mathbb{M}_{n_2}$ is $(c_1 = \text{Enc}(n_1, m, r_1), c_2 = \text{Enc}(n_2, m, r_2), \pi)$ where π is a proof that c_1 and c_2 encrypt the same plaintext. Now assume (as typically the case with single-key factoring-based encryption) there exists an efficient way to generate a random-looking c_1 such that its decryption $\text{Dec}(\text{factors}(n_1), c_1)$ leads to an immediate recovery of $\text{factors}(n_1)$. In a typical reduction \mathcal{R} from $\text{FACT}[\mathcal{E}.\text{keygen}]$ to breaking the OW-CPA security of $\text{NY}(\mathcal{E})$, \mathcal{R} takes as input a modulus $n_1 \leftarrow \mathcal{E}.\text{keygen}(1^k)$ but generates by itself the second key pair $(n_2, \text{factors}(n_2)) \leftarrow \mathcal{E}.\text{keygen}(1^k)$ and r to constitute a public key $\text{pk} = (n_1, n_2, r)$. Since \mathcal{R} fully controls the generation of n_2 and r , \mathcal{R} can use the simulator of the underlying NIZK proof system to create a valid encryption $c = (c_1, c_2, \pi)$ for a random c_1 . Calling the OW-CPA adversary will then provide $\text{Dec}(\text{factors}(n_1), c_1)$, thus allowing \mathcal{R} to factor n_1 . The meta-reduction \mathcal{M} playing the UBK-CCA game against $\text{NY}(\mathcal{E})$ however, is given some public key $\text{PK} = (N_1, N_2, R)$ and a decryption oracle implicitly parameterized by PK . Since \mathcal{R} takes as input a single modulus and generates by itself the rest of the public key to be given to its adversarial oracle, \mathcal{M} cannot, even if \mathcal{R} is run on input N_1 , use the decryption oracle to answer the request(s) $((N_1, n_2, r), (c_1, c_2, \pi))$ made by \mathcal{R} because $\Pr[n_2 \neq N_2 \vee r \neq R]$ is overwhelming.

6 Are Key Generators Non-malleable?

Our extended impossibility results apply to single-key encryption schemes based on non-malleable key generation. We conjecture that most instance generators are in turn non-malleable and expect to see further research works based on this property in the future. A possible improvement of this work would be to give a formal proof of non-malleability for commonly referred generators such as RSA-3 or Sophie-Germain using computational number theory. Another issue is the design of non-trivial examples of *malleable* key generators.

Acknowledgements. We thank the anonymous referees of Asiacrypt'06 for their numerous comments as well as Mihir Bellare for suggestions that substantially improved the presentation of this paper. This work has been financially supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

1. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, LNCS 1462, pp. 26–46. Springer Verlag, 1998.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, pp. 62–73, 1993.

3. M. Bellare and P. Rogaway. Optimal asymmetric encryption: How to encrypt with RSA. In *EUROCRYPT'94*, LNCS 950, pp. 92–111. Springer Verlag, 1994.
4. M. Blum and S. Goldwasser. An efficient probabilistic public key encryption scheme which hides all partial information. In *CRYPTO'84*, LNCS 196, pp. 289–299, 1985.
5. D. Boneh. Simplified OAEP for the RSA and Rabin functions. In *CRYPTO'01*, LNCS 2139, pp. 275–291. Springer Verlag, 2001.
6. D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring (extended abstract). In *EUROCRYPT'98*, LNCS 1403, pp. 59–71, 1998.
7. D. R. L. Brown. Unprovable security of RSA-OAEP in the standard model, 2006. <http://eprint/iacr.org/2006/223>.
8. B. Chor and O. Goldreich. RSA/Rabin least significant bits are $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$ secure. In *CRYPTO'84*, LNCS 196, pp. 303–313. Springer Verlag, 1985.
9. N. Demytko. A new elliptic curve based analogue of RSA. In *EUROCRYPT'93*, LNCS 765, pp. 40–49. Springer Verlag, 1994.
10. D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. In *ACM STOC'91*, pp. 542–552, 1991.
11. E. Fujisaki. Chosen-chiphertext security of EPOC-2. Technical report, NTT Corporation, 2001.
12. E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, S. Okazaki, D. Pointcheval and S. Uchiyama. EPOC: Efficient probabilistic public-key encryption. Submitted to ISO and NESSIE.
13. S. Goldwasser, S. Micali and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. on Comp.*, 17(2):281–308, April 1988.
14. D. Jao, S. Miller and R. Venkatesan. Do all elliptic curves of the same order have the same difficulty of disc. log? In *ASIACRYPT'05*, LNCS 3788, pp. 21–40, 2005.
15. K. Koyama, U. Maurer, T. Okamoto and S. Vanstone. New public-key schemes based on curves over the ring \mathbb{Z}_n . In *CRYPTO'91*, LNCS 576, pp. 252–266, 1992.
16. T. Malkin, R. Moriarty and N. Yakovenko. Generalized environmental security from number theoretic assumptions. In *TCC'06*, LNCS 3876, pp. 343–359, 2006.
17. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen-ciphertext attacks. In *22nd ACM Symposium on Theory of Computing*, 1990.
18. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT-RSA'01*, LNCS 2020, pp. 159–175, 2001.
19. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, LNCS 1592, pp. 223–238. Springer Verlag, 1999.
20. M. O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Jan. 1979.
21. RSA Data Security. PKCS #1: RSA encryption standard, Nov. 1993. Version 1.5.
22. H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, IT-26(6):726–729, 1980.

A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants

Ellen Jochemsz^{1,*} and Alexander May²

¹ Department of Mathematics and Computer Science,
TU Eindhoven, 5600 MB Eindhoven, The Netherlands
e.jochemsz@tue.nl

² Faculty of Computer Science
TU Darmstadt, 64289 Darmstadt, Germany
may@informatik.tu-darmstadt.de

Abstract. We describe a strategy for finding small modular and integer roots of multivariate polynomials using lattice-based Coppersmith techniques. Applying our strategy, we obtain new polynomial-time attacks on two RSA variants. First, we attack the Qiao-Lam scheme that uses a Chinese Remaindering decryption process with a small difference in the private exponents. Second, we attack the so-called Common Prime RSA variant, where the RSA primes are constructed in a way that circumvents the Wiener attack.

Keywords: lattices, small roots, Coppersmith's method, RSA variants, cryptanalysis.

1 Introduction

Since Coppersmith introduced new ways of finding small modular and integer roots of polynomials in 1996 [4,5,6], variations of these methods have been widely used in the field of cryptanalysis. Let us give an example that demonstrates the usefulness of computing small roots. In the case of RSA, the public variables (N, e) and the secret variables (d, p, q) satisfy the relation

$$ed - 1 = k(N - (p + q - 1)), \text{ for some (unknown) } k.$$

It is known that one can use Coppersmith techniques to try to find the integer root $(d, k, p + q - 1)$ of the polynomial $f(x, y, z) = ex - yN + yz - 1$, and hence recover the factorization of N . Alternatively, one could look for the modular root $(k, p + q - 1)$ of $f_e(y, z) = y(N - z) + 1$ modulo e .

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

The success of the application of a Coppersmith technique depends on the size of the root. More precisely, the analysis of the attack results in a bound on the size of roots that can be found with this method in polynomial time. For the case of finding the root $(y^{(0)}, z^{(0)}) = (k, p + q - 1)$ of $f_e(y, z) = y(N - z) + 1$ modulo e in the example above, Boneh and Durfee [1] used a Coppersmith technique to obtain the bound

$$Y^{2+3\tau} Z^{1+3\tau+3\tau^2} < e^{1+3\tau}, \text{ for } |y^{(0)}| < Y, \text{ and } |z^{(0)}| < Z,$$

where $\tau > 0$ can be optimized once the sizes of Y , Z , and e are known. This led Boneh and Durfee to show that for $d < N^{0.284}$ the secret RSA parameters can be recovered in polynomial time, which they later refined to $d < N^{0.292}$ in the same work [1].

Since the analysis of a polynomial f of which we wish to find a small root heavily depends on the monomials that appear in f , each new polynomial has to be analyzed anew. This is typically a tedious and non-trivial task. In 2005, Blömer and May [3] showed how to find optimal bounds for small integer roots of bivariate polynomials. In this paper we present a heuristic strategy that applies to all multivariate polynomials; having either modular or integer roots.

We apply our strategy to derive new heuristic attacks on two RSA variants, using a polynomial that arises in their cryptanalysis. In the first system, the Chinese Remainder Theorem is used in the decryption phase, with the special property that $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$ have a fixed difference $d_p - d_q$. This scheme was proposed in 1998 by Qiao and Lam [17] who suggested to use the small difference $d_p - d_q = 2$. The benefit of the Qiao-Lam scheme is that one has to store only one out of the two keys d_p, d_q and the small difference itself. Up to now, the best attack on the Qiao-Lam scheme was a meet-in-the-middle attack with time and space complexity $\tilde{O}\{\sqrt{\min\{d_p, d_q\}}\}$ [17].

Qiao and Lam proposed to use a 1024-bit modulus N with 128-bit d_p, d_q . Moreover, they argued that in practice 96-bit private exponents should provide sufficient security. Our results show that private exponents up to $N^{0.099}$ can be recovered in polynomial time. Hence, for 1024-bit RSA moduli one can recover 96-bit d_p, d_q in polynomial time. Furthermore, attacking 128-bit private exponents should also be feasible with our attack by adding some brute force search on the most significant bits. We confirm the validity of our heuristic attack by providing several experiments. Although recovering 96-bit private exponents can theoretically be done in polynomial time, in practice it turns out to be a non-trivial task since it requires an LLL-lattice basis reduction [13] in large dimension.

We would like to point out that our attack works whenever $\max\{d_p, d_q\} \leq N^{0.099-\epsilon}$ for some arbitrarily small constant ϵ , and the difference $d_p - d_q$ is known to the attacker. We do not require that the difference $d_p - d_q$ itself is a small constant like in the Qiao-Lam scheme.

As a second application of our strategy, we give a new attack on an RSA variant called Common Prime RSA. This variant was originally proposed by Wiener [19] as a countermeasure for his attack on small secret exponents $d \leq N^{\frac{1}{4}}$.

The suggestion is to choose p, q such that $p - 1$ and $q - 1$ share a large gcd. In 1995, Lim and Lee [12] used this Common Prime RSA variant in a server-aided RSA protocol, which was attacked in 1998 by McKee and Pinch [15]. Recently, Hinek [9] revisited the Common Prime RSA variant. He proposed several RSA parameter settings with secret exponents less than $N^{\frac{1}{4}}$. However, our second heuristic attack shows that parts of the proposed key space lead to polynomial time attacks on RSA. We demonstrate the practicality of our second attack by providing several experiments that recover the RSA secret information.

2 Finding Small Roots

In this section we describe some tools that we use to solve the problem of finding small roots, for both the modular and the integer case. Moreover, we present our new strategy.

In [4,5,6], Coppersmith describes rigorous techniques to find small integer roots of polynomials in a single variable modulo N , and polynomials in two variables over the integers. The methods extend to more variables, making them heuristical. Howgrave-Graham reformulated Coppersmith’s ideas of finding modular roots in [11], of which we use the following (generalized) lemma.

Lemma 1 (Howgrave-Graham). *Let $h(x_1, \dots, x_n) \in \mathbb{Z}[x_1, \dots, x_n]$ be an integer polynomial that consists of at most ω monomials. Suppose that*

- (1) $h(x_1^{(0)}, \dots, x_n^{(0)}) \equiv 0 \pmod N$ for some $|x_1^{(0)}| < X_1, \dots, |x_n^{(0)}| < X_n$, and
- (2) $\|h(x_1 X_1, \dots, x_n X_n)\| < \frac{N}{\sqrt{\omega}}$.

Then $h(x_1^{(0)}, \dots, x_n^{(0)}) = 0$ holds over the integers.

In Lemma 1 the norm of a polynomial $f(x_1, \dots, x_n) = \sum a_{i_1 \dots i_n} x_1^{i_1} \dots x_n^{i_n}$ is the Euclidean norm of its coefficient vector: $\|f(x_1, \dots, x_n)\|^2 := \sum |a_{i_1 \dots i_n}|^2$.

Howgrave-Graham’s lemma is usually combined with LLL reduction of lattice bases, designed by Lenstra, Lenstra, and Lovász [13]. A proof of the following fact can be found in [14].

Fact 1 (LLL). *Let L be a lattice of dimension ω . In polynomial time, the LLL-algorithm outputs reduced basis vectors $v_i, 1 \leq i \leq \omega$ that satisfy*

$$\|v_1\| \leq \|v_2\| \leq \dots \leq \|v_i\| \leq 2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} \det(L)^{\frac{1}{\omega+1-i}}.$$

Thus the condition $2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} \det(L)^{\frac{1}{\omega+1-i}} < \frac{N}{\sqrt{\omega}}$ implies that the polynomials corresponding to the shortest i reduced basis vectors match Howgrave-Graham’s bound. This reduces to

$$\det(L) \leq 2^{\frac{-\omega(\omega-1)}{4}} \left(\frac{1}{\sqrt{\omega}}\right)^{\omega+1-i} N^{\omega+1-i}.$$

In the analysis, we let terms that do not depend on N contribute to an error term ϵ , and simply use the determinant condition $\det(L) \leq N^{\omega+1-i}$.

2.1 Strategy for Finding Small Modular Roots

We will now describe our strategy to find small modular roots of polynomials. Suppose we want to find a small root $(x_1^{(0)}, \dots, x_n^{(0)})$ of a polynomial f_N modulo a known composite integer N of unknown factorization. We assume that we know an upper bound for the root, namely $|x_j^{(0)}| < X_j$ for some given X_j , for $j = 1, \dots, n$.

Let l be a leading monomial of f_N , with coefficient a_l . That is, there is no monomial in f_N besides l that is divisible by l . Then $\gcd(N, a_l)$ is 1, or else we have found a factor of N . Therefore, we can use $f'_N = a_l^{-1}f_N \pmod N$.

We start by explaining the basic strategy to find the small modular roots, after which we extend it slightly to obtain the full strategy.

Basic Strategy: Let $\epsilon > 0$ be an arbitrarily small constant. Depending on $\frac{1}{\epsilon}$, we fix an integer m . For $k \in \{0, \dots, m + 1\}$, we define the set M_k of monomials

$$M_k := \{x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \mid x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \text{ is a monomial of } f_N^m \text{ and } \frac{x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}}{l^k} \text{ is a monomial of } f_N^{m-k}\}.$$

In this definition of M_k and throughout this paper, we assume that the monomials of f_N, \dots, f_N^{m-1} are all contained in the monomials of f_N^m . If this is not the case, the definition can be slightly changed such that M_k contains all monomials $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}$ of f_N^j for $j \in \{1, \dots, m\}$ for which $\frac{x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}}{l^k}$ is a monomial of f_N^j for some $i \in \{0, \dots, m - k\}$. Notice that by definition the set M_0 contains all the monomials in f_N^m , whereas $M_{m+1} = \emptyset$.

Next, we define the following shift polynomials:

$$g_{i_1 \dots i_n}(x_1, \dots, x_n) := \frac{x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}}{l^k} f'_N(x_1, \dots, x_n)^k N^{m-k},$$

for $k = 0, \dots, m$, and $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in M_k \setminus M_{k+1}$.

All polynomials g have the root $(x_1^{(0)}, \dots, x_n^{(0)})$ modulo N^m . We define a lattice L by taking the coefficient vectors of $g(x_1 X_1, \dots, x_n X_n)$ as a basis. We can force the matrix describing L to be lower triangular, if we use the following ordering of the columns of the matrix. A column corresponding to the monomial $x_1^{i_1} \dots x_n^{i_n} \in M_k \setminus M_{k+1}$ has smaller order than a column corresponding to $x_1^{j_1} \dots x_n^{j_n} \in M_{k'} \setminus M_{k'+1}$ if $k < k'$. If $k' = k$, then a lexicographical ordering of the monomials is used. The columns in the lattice basis appear in increasing order from left to right. The diagonal elements are those corresponding to the monomial l^k in $(f'_N)^k$ for each row. Therefore, the diagonal terms of the matrix are $X_1^{i_1} X_2^{i_2} \dots X_n^{i_n} N^{m-k}$ for the given combinations of k and i_j .

The intuition behind the choice of the sets M_k can be explained as follows. We aim to have a matrix with a low determinant. To keep the diagonal element corresponding to the monomial $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}$ of f_N^m as small as possible, we use the largest possible powers of f_N in the shifts. The condition that $\frac{x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}}{l^k}$ is a monomial of f_N^{m-k} ensures that no monomials appear that are not in f_N^m .

For a small example, consider the polynomial $f_N(x, y) = 1 + xy^2 + x^2y$. Let us take $l = x^2y$ as our leading term, and $m = 2$. We want to build a lattice whose columns correspond to the monomials $\{1, xy^2, x^2y, x^2y^4, x^3y^3, x^4y^2\}$ of f_N^2 . The shifts given by our strategy are:

$$\begin{array}{ll} \text{for } 1 \in M_0 \setminus M_1: N^2 & \text{for } x^2y \in M_1 \setminus M_2: f_N N \\ \text{for } xy^2 \in M_0 \setminus M_1: xy^2 N^2 & \text{for } x^3y^3 \in M_1 \setminus M_2: xy^2 f_N N \\ \text{for } x^2y^4 \in M_0 \setminus M_1: x^2y^4 N^2 & \text{for } x^4y^2 \in M_2 \setminus M_3: f_N^2 \end{array}$$

Note that the monomial x^2y^4 is not in M_1 . Although x^2y^4 is divisible by $l = x^2y$ and therefore we could obtain x^2y^4 also by using the shift $y^3 f_N N$, the product $y^3 f_N$ would produce the new monomials y^3 and xy^5 , which are not in f_N^2 .

In general, we find that our condition $\det(L) < N^{m(\omega+1-n)}$, derived from Lemma 1 and Fact 1, reduces to

$$\prod_{j=1}^n X_j^{s_j} < N^{s_N}, \text{ for } \begin{cases} s_j = \sum_{x_n^{i_1} \dots x_n^{i_n} \in M_0} i_j, \text{ and} \\ s_N = \sum_{k=0}^m k(|M_k| - |M_{k+1}|) = \sum_{k=1}^m |M_k| \end{cases} \quad (1)$$

If we follow this procedure for a given f_N , then (1) will give us an upper bound on the size of the root that we are trying to find. For X_j and N satisfying this bound we obtain n polynomials h_i such that $h_i(x_1^{(0)}, \dots, x_n^{(0)}) = 0$. If the polynomials h_i are algebraically independent, i.e. they do not share a non-trivial gcd, then resultant computations will reveal the root. Under Assumption 1 this will lead us to finding $(x_1^{(0)}, \dots, x_n^{(0)})$.

Assumption 1. *The resultant computations for the polynomials h_i yield non-zero polynomials.*

All methods for $n \geq 2$ have a similar assumption concerning the algebraic independence of the polynomials h_i . Therefore one has to keep in mind that (most) attacks using Coppersmith techniques are heuristical, and experiments must be done for specific cases to justify the assumption.

Extended Strategy: For many polynomials, it is profitable to use extra shifts of a certain variable. For instance, if we use extra shifts of x_1 , then we can extend our basic strategy by using

$$M_k := \bigcup_{0 \leq j \leq t} \{x_1^{i_1+j} x_2^{i_2} \dots x_n^{i_n} \mid x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \text{ is a monomial of } f_N^m \text{ and } \frac{x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}}{l^k} \text{ is a monomial of } f_N^{m-k}\}.$$

Moreover, extra shifts of several variables, or combined shifts should be considered to obtain an optimal bound.

Using this new definition of M_k , the rest of the strategy conforms to the basic strategy as described before. In Appendix A, we show how the known results on small modular roots from [1,2,6] are all special cases of our basic or extended strategy.

2.2 Strategy for Finding Small Integer Roots

Coron reformulated Coppersmith’s method of finding small integer roots in [7]. Essentially, Coron picks a ‘suitable’ integer R and transforms the situation into finding a small root modulo R , after which one can apply Howgrave-Graham’s lemma. Analogous to Coron, we will now present our heuristic strategy for finding small integer roots of multivariate polynomials. The result is an extension of the result given by Blömer and May [3], that was meant for the provable special case of bivariate polynomials.

We note that one could also use Coppersmith’s original technique instead of Coron’s reformulation. The advantage to do so is that in the original Coppersmith technique, lattices of smaller dimension are required. The asymptotic bounds obtained by both methods are equivalent, but the difference is in the size of the error term ϵ . For this paper, we have chosen to use Coron’s method for the sake of a simpler notation, an easier implementation and for its similarity to the modular approach.

Suppose we want to find the small integer root $(x_1^{(0)}, \dots, x_n^{(0)})$ of an irreducible polynomial f . We know that the root is small in the sense that $|x_j^{(0)}| < X_j$, for $j = 1, \dots, n$.

Analogous to Section 2.1, we fix an integer m depending on $\frac{1}{\epsilon}$. We call d_j the maximal degree of x_j in f , and W the maximal coefficient of $f(x_1X_1, \dots, x_nX_n)$. We will use $W = \|f(x_1X_1, \dots, x_nX_n)\|_\infty$, with $\|f(x_1, \dots, x_n)\|_\infty := \max |a_{i_1 \dots i_n}|$ for $f(x_1, \dots, x_n) = \sum a_{i_1 \dots i_n} x_1^{i_1} \dots x_n^{i_n}$ as a notation. Moreover, we define $R = W \prod_{j=1}^n X_j^{d_j(m-1)}$. To work with a polynomial with constant term 1, we define $f' = a_0^{-1} f \bmod R$, where a_0 is the constant term of f . This means that we should have $a_0 \neq 0$ and $\gcd(a_0, R) = 1$. The latter is easy to achieve, analogous to [7, Appendix A], since any X_j with $\gcd(a_0, X_j) \neq 1$ can be changed into an X'_j such that $X_j < X'_j < 2X_j$ and $\gcd(a_0, X'_j) = 1$. The same holds for W .

Let us now consider the case $a_0 = 0$. In [7, Appendix A], Coron discussed this case for bivariate polynomials, and showed a simple way to transfer a polynomial f with zero constant term into a polynomial f^* with non-zero constant term.

A general way to do this for multivariate polynomials would be the following. First, we find a non-zero integer vector (y_1, \dots, y_n) such that $f(y_1, \dots, y_n) \neq 0$. This can be constructed in polynomial time since there are only polynomially many roots within the given bounds. Then we define $f^*(x_1, \dots, x_n) := f(x_1 + y_1, \dots, x_n + y_n)$, and look for roots of f^* . Since $f^*(0, \dots, 0) = f(y_1, \dots, y_n)$, f^* has a non-zero constant term.

We would like to point out that the switch to f^* will affect the set of monomials, and new monomials may appear in f^* that were not in f . This may affect the analysis and lead to a different Coppersmith-type bound. This issue already appears with bivariate polynomials, but it did not affect Coron’s analysis since in his case the set of monomials stayed the same.

Let us now describe our strategy for finding integer roots. As before, we start with the basic strategy, that we extend later to obtain the full strategy.

Basic Strategy: Let us first fix an arbitrarily small error term ϵ . We define an integer m depending on $\frac{1}{\epsilon}$. Furthermore, we define the sets S and M of monomials that represent the monomials of f^{m-1} and f^m respectively. We denote by l_j the largest exponent of x_j that appears in the monomials of S , i.e. $l_j = d_j(m - 1)$.

Next, we define the following shift polynomials

$$g : x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} f'(x_1, \dots, x_n) \prod_{j=1}^n X_j^{l_j - i_j} \quad , \text{ for } x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in S,$$

$$g' : x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} R \quad , \text{ for } x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in M \setminus S.$$

All g and g' have the root $(x_1^{(0)}, \dots, x_n^{(0)})$ modulo R . The coefficient vectors of $g(x_1 X_1, \dots, x_n X_n)$ and $g'(x_1 X_1, \dots, x_n X_n)$ form a lattice basis of a lattice L .

Using lexicographical ordering of the monomials, we can order the basis matrix such that it is upper triangular. The diagonal elements of the rows of g are those corresponding to the constant term in f' . Therefore, the diagonal entries of the matrix are $\prod_{j=1}^n X_j^{d_j(m-1)}$ for the polynomials g and $W \prod_{j=1}^n X_j^{d_j(m-1) + i_j}$ for the polynomials g' .

From Section 2, we know that the determinant condition $\det(L) < R^{\omega+2-n}$ ensures that the $n - 1$ smallest vectors in an LLL reduced basis of L correspond to $n - 1$ polynomials $h_i(x_1, \dots, x_n)$ with $h_i(x_1^{(0)}, \dots, x_n^{(0)}) = 0$.

We find that the condition $\det(L) < R^{\omega+2-n}$ reduces to

$$\prod_{j=1}^n X_j^{s_j} < W^{s_W}, \text{ for } s_j = \sum_{x_1^{i_1} \dots x_n^{i_n} \in M \setminus S} i_j, \text{ and } s_W = |S|. \tag{2}$$

So if (2) holds, we obtain $n - 1$ polynomials h_i such that $h_i(x_1^{(0)}, \dots, x_n^{(0)}) = 0$.

The choice of R ensures that the h_i are independent of f . This is because all h_i are divisible by $\prod_{j=1}^n X_j^{d_j(m-1)}$. According to a generalization by Hinek/Stinson [10, Corollary 5] of a lemma of Coron [7], a multiple $h(x_1, \dots, x_n)$ of $f(x_1, \dots, x_n)$ that is divisible by $\prod_{j=1}^n X_j^{d_j(m-1)}$ has norm at least

$$2^{-(\rho+1)^n + 1} \prod_{j=1}^n X_j^{d_j(m-1)} W = 2^{-(\rho+1)^n + 1} R,$$

where ρ is the maximum degree of the polynomials f, h in each variable separately. If we let terms that do not depend on R contribute to ϵ , it follows that if h_i satisfies Howgrave-Graham's bound $\|h_i(x_1 X_1, \dots, x_n X_n)\| < \frac{R}{\sqrt{\omega}}$, then it also cannot be a multiple of f . Since we assume that f is irreducible, it follows that f and h_i must be algebraically independent. However we cannot prevent that the h_i are pairwise algebraically dependent. So the resultant computations of f and h_i (for $i = 1, \dots, n - 1$) will only reveal the root under Assumption 1. This makes the techniques heuristical for $n \geq 3$.

Extended Strategy: As in the modular case, our strategy is not finished before exploring the possibilities of extra shifts of a certain variable (or more variables).

Suppose we use extra shifts of the variable x_1 . Then, instead of $S = \{\text{monomials of } f^{m-1}\}$, and $M = \{\text{monomials of } f^m\}$, we use

$$S = \bigcup_{0 \leq j \leq t} \{x_1^{i_1+j} x_2^{i_2} \dots x_n^{i_n} \mid x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \text{ is a monomial of } f^{m-1}\},$$

$$M = \{\text{monomials of } x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \cdot f \mid x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in S\}.$$

With the new definitions, the rest of the strategy conforms to the basic strategy as described above, except for the value of R . It is necessary to change $R = W \prod_{j=1}^n X_j^{d_j(m-1)}$ into $R = W \prod_{j=1}^n X_j^{l_j}$, where l_j is the largest exponent of x_j that appears in the monomials of S . In Appendix B, we show that the known results on small integer roots from [3,6,8] are special cases of our basic or extended strategy. Moreover, a detailed example for a specific polynomial is treated in the next section.

3 A Bound Obtained with the New Strategy

In this section we will give a novel analysis of a trivariate polynomial that will be used in two new attacks on RSA variants in the subsequent sections.

Let $f(x, y, z) = a_0 + a_1x + a_2x^2 + a_3y + a_4z + a_5xy + a_6xz + a_7yz$ be a polynomial with a small root $(x^{(0)}, y^{(0)}, z^{(0)})$, with $|x^{(0)}| < X, |y^{(0)}| < Y, |z^{(0)}| < Z$. We show that under Assumption 1 for every fixed ϵ , all sufficiently small roots can be found in time polynomial in $\log W$ provided that

$$X^{7+9\tau+3\tau^2} (YZ)^{5+\frac{9}{2}\tau} < W^{3+3\tau-\epsilon},$$

where we can optimize $\tau > 0$ after the substitution of values for X, Y, Z , and W .

Let us follow the extended strategy described in Section 2.2 to show how this bound can be obtained. Our goal is to construct two polynomials h_1, h_2 with the root $(x^{(0)}, y^{(0)}, z^{(0)})$ that are not multiples of f . To do so, we fix an integer m depending on $\frac{1}{\epsilon}$, and an integer $t = \tau m$ that describes the number of extra x -shifts. We define $R = WX^{2(m-1)+t}(YZ)^{m-1}$ and $f' = a_0^{-1}f \pmod R$. The shift polynomials g and g' are given by:

$$g : x^{i_1} y^{i_2} z^{i_3} f'(x, y, z) X^{2(m-1)+t-i_1} Y^{m-1-i_2} Z^{m-1-i_3} \text{ for } x^{i_1} y^{i_2} z^{i_3} \in S,$$

$$g' : R x^{i_1} y^{i_2} z^{i_3} \text{ for } x^{i_1} y^{i_2} z^{i_3} \in M \setminus S,$$

for

$$S = \bigcup_{0 \leq j \leq t} \{x_1^{i_1+j} y^{i_2} z^{i_3} \mid x_1^{i_1} y^{i_2} z^{i_3} \text{ is a monomial of } f^{m-1}\},$$

$$M = \{\text{monomials of } x_1^{i_1} y^{i_2} z^{i_3} \cdot f \mid x_1^{i_1} y^{i_2} z^{i_3} \in S\}.$$

It follows that

$$x^{i_1} y^{i_2} z^{i_3} \in S \Leftrightarrow i_2 = 0, \dots, m-1; i_3 = 0, \dots, m-1;$$

$$i_1 = 0, \dots, 2(m-1) - (i_2 + i_3) + t.$$

$$x^{i_1} y^{i_2} z^{i_3} \in M \Leftrightarrow i_2 = 0, \dots, m; i_3 = 0, \dots, m; i_1 = 0, \dots, 2m - (i_2 + i_3) + t.$$

All polynomials g and g' have the root $(x^{(0)}, y^{(0)}, z^{(0)})$ modulo R . Let h_1 and h_2 be linear combinations of the polynomials g and g' . As was explained in Section 2.2,

if h_1 and h_2 satisfy Howgrave-Graham's bound $\|h_i(xX, yY, zZ)\| < \frac{R}{\sqrt{\omega}}$, then we can assume that h_1 and h_2 both have the root $(x^{(0)}, y^{(0)}, z^{(0)})$ over the integers, and also that they are algebraically independent of f .

Using the coefficient vectors of $g(xX, yY, zZ)$ and $g'(xX, yY, zZ)$ as a basis, we build a lattice L . We order the vectors such that the matrix is triangular, with the diagonal entries of g equal to $X^{2(m-1)+t}(YZ)^{m-1}$, and those of g' equal to $RX^{i_1}Y^{i_2}Z^{i_3} = X^{2(m-1)+t+i_1}Y^{m-1+i_2}Z^{m-1+i_3}W$.

Now by (2), provided that $\prod_{j=1}^n X_j^{s_j} < W^{|S|}$ with $s_j = \sum_{x_1^{i_1} \dots x_n^{i_n} \in M \setminus S} i_j$ holds, the polynomials h_1 and h_2 corresponding to the shortest two LLL-reduced basis vectors satisfy Howgrave-Graham's bound. This reduces to

$$X^{(\frac{7}{3}+3\tau+\tau^2)m^3+o(m^2)}(YZ)^{(\frac{5}{3}+\frac{3}{2}\tau)m^3+o(m^2)} \leq W^{(1+\tau)m^3+o(m^2)}.$$

If we let all terms of order $o(m^2)$ contribute to ϵ , the condition simplifies to

$$X^{7+9\tau+3\tau^2}(YZ)^{5+\frac{9}{2}\tau} < W^{3+3\tau-\epsilon}.$$

4 Attack on RSA-CRT with Known Difference

In this section, we explain how a small root of a polynomial $f(x, y, z) = a_0 + a_1x + a_2x^2 + a_3y + a_4z + a_5xy + a_6xz + a_7yz$ results in a new attack on a variant of RSA-CRT proposed by Qiao/Lam [17]. We show the following result.

Theorem 1 (RSA-CRT with Fixed Known Difference $d_p - d_q$)

Under Assumption 1, for every $\epsilon > 0$, there exists n_0 such that for every $n > n_0$, the following holds: Let $N = pq$ be an n -bit RSA modulus, and p, q primes of bitsize $\frac{n}{2}$. Let $ed \equiv 1 \pmod{\phi(N)}$, and d_p and d_q be such that $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$. Assume that d_p and d_q are chosen such that $d_p = d_q + \bar{c}$ for some known \bar{c} and $\text{bitsize}(d_p), \text{bitsize}(d_q) \leq \delta n$ for some $0 < \delta < \frac{1}{2}$. Then N can be factored in time polynomial in $\log N$ provided that

$$\delta < \frac{1}{4}(4 - \sqrt{13}) - \epsilon.$$

Notice that $\frac{1}{4}(4 - \sqrt{13}) \approx 0.099$. Hence, our attack applies whenever d_p or d_q is smaller than $N^{0.099-\epsilon}$ and the difference $\bar{c} = d_p - d_q$ is known to an attacker.

4.1 RSA-CRT with Known Difference $d_p - d_q$

In 1990, Wiener [19] showed that choosing $d < N^{\frac{1}{4}}$ makes RSA insecure. As an alternative, Wiener suggested to use the Chinese Remainder Theorem (CRT) for the decryption phase of RSA: Instead of computing $m \equiv c^d \pmod{N}$ for some ciphertext c , compute $m_1 \equiv c^{d_p} \pmod{p}$ and $m_2 \equiv c^{d_q} \pmod{q}$ and then combine these results using CRT to obtain m . Wiener pointed out that both exponents $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$ could be chosen small to obtain a fast decryption. Then usually e is of the same size as the modulus N .

Qiao and Lam [17] proposed to use d_p and d_q such that $d_p - d_q = 2$ in their method for fast signature generation on a low-cost smartcard. For the size of d_p and d_q , they suggest 128 bits to counteract the birthday attack that they describe in [17]. Moreover, they state that 96 bits should be enough to counteract this attack in practice. In current proposals, a minimum of 160 bits is advised for the private exponents to counteract the birthday attack.

4.2 Description of the New Attack

When $d_p - d_q = \bar{c}$, the public and private variables of RSA-CRT satisfy the following relations.

$$\begin{cases} ed_p = 1 + k(p - 1), \\ e(d_p - \bar{c}) = 1 + l(q - 1), \end{cases} \quad \text{or equivalently} \quad \begin{cases} ed_p - 1 + k = kp, \\ ed_p - \bar{c}e - 1 + l = lq. \end{cases}$$

Multiplying the two equations results in

$$(1 + \bar{c}e) - (2e + \bar{c}e^2)d_p + e^2d_p^2 - (\bar{c}e + 1)k - l + ed_pk + ed_pl + (1 - N)kl = 0,$$

in which the unknowns are d_p , k , and l . We can extract from this equation that

$$f(x, y, z) = (1 + \bar{c}e) - (2e + \bar{c}e^2)x + e^2x^2 - (\bar{c}e + 1)y - z + exy + exz + (1 - N)yz$$

has a small root (d, k, l) . From (d, k, l) , the factorization of N can easily be found. Suppose $\max\{d_p, d_q\}$ is of size N^δ for some $\delta \in (0, \frac{1}{2})$. Then k and l are both bounded by $N^{\delta + \frac{1}{2}}$ (here we omit constants and let these contribute to the error term ϵ). Therefore, we put $X = N^\delta$, $Y = Z = N^{\delta + \frac{1}{2}}$, and $W = N^{2+2\delta}$.

In Section 3 we showed that for this polynomial, the asymptotic bound is

$$X^{7+9\tau+3\tau^2}(YZ)^{5+\frac{9}{2}\tau} < W^{3+3\tau},$$

where $\tau > 0$ can be optimized. Substituting the values for X , Y , Z , and W , we obtain

$$(7 + 9\tau + 3\tau^2)\delta + (5 + \frac{9}{2}\tau)(2\delta + 1) - (3 + 3\tau)(2\delta + 2) < 0, \text{ or}$$

$$3\delta\tau^2 + 3(4\delta - \frac{1}{2})\tau + (11\delta - 1) < 0.$$

For the optimal value $\tau = \frac{\frac{1}{2}-4\delta}{2\delta}$, this reduces to $\delta < \frac{1}{4}(4 - \sqrt{13}) \approx 0.099$.

Therefore, for a 1024 bit modulus N , the system should be considered unsafe when d_p is at most $0.099 \cdot 1024 \approx 101$ bits. This breaks the system of Qiao and Lam for the proposed 96 bit exponents in time polynomial in the bit-size of N .

We can add an exhaustive search on the most significant bits of d_p and try the attack for each candidate for \tilde{d}_p . Here, $d_p = \tilde{d}_p + d_0$, where the unknown part of d is d_0 . The corresponding polynomial f will change, but it will still have the same monomials. Therefore, the analysis will follow easily. The proposal of Qiao and Lam to use 128 bit private exponents can also be considered unsafe when applying

such an extra exhaustive search, although performing such an attack may be costly in practice.

We performed several experiments to test the validity of Assumption 1 and to show which results can be achieved with relatively small lattices. We implemented the new attacks on a 2.4GHz Pentium running Linux. The LLL lattice reduction was done using Shoup’s NTL [18]. For the attack on RSA-CRT with known difference described in Section 4, the parameters d_p, d_q were chosen with difference $d_p - d_q = 2$ as suggested in the Qiao-Lam scheme. For $m = 2$ the choice $t = 8$ maximizes the size of the attackable d_p .

N	d_p	lattice parameters	LLL-time
1000 bit	10 bit	$m = 2, t = 3, \dim = 54$	32 min
2000 bit	22 bit	$m = 2, t = 3, \dim = 54$	175 min
3000 bit	42 bit	$m = 2, t = 3, \dim = 54$	487 min
4000 bit	60 bit	$m = 2, t = 3, \dim = 54$	1015 min
5000 bit	85 bit	$m = 2, t = 3, \dim = 54$	1803 min
500 bit	9 bit	$m = 2, t = 8, \dim = 99$	105 min
1000 bit	18 bit	$m = 2, t = 8, \dim = 99$	495 min
500 bit	13 bit	$m = 3, t = 3, \dim = 112$	397 min

In each experiment we obtained two polynomials $h_1(x, y, z), h_2(x, y, z)$ with the desired root $(x^{(0)}, y^{(0)}, z^{(0)})$. Solving $g(z) = \text{Res}_y(\text{Res}_x(h_1, f), \text{Res}_x(h_2, f)) = 0$ yielded the unknown $z^{(0)}$. The parameters $y^{(0)}$ and $x^{(0)}$ could be obtained by back substitution. The resultant heuristic of Assumption 1 worked perfectly in practice. For every instance, we could recover the secrets and hence factor N .

One should note that our experiments are quite far from solving the proposed 96-bit d_p, d_q instances of the Qiao-Lam scheme. Theoretically, the smallest m for which one obtains the 96-bit bound is $m = 61$ together with $t = 36$, resulting in a lattice dimension of 376712. Reducing lattice bases in this dimension is clearly out of reach.

However, we would like to point out that we did not optimize the performance of our attack. For optimization of the running-time, one should combine brute-force guessing of most significant bits of d_p with the described lattice attack. Moreover, one should apply faster lattice reduction methods like the recently proposed L^2 -method of Nguyen, Stehlé [16]. Additionally, a significant practical improvement should be obtained by implementing Coppersmith’s original method instead of Coron’s method, since in Coppersmith’s method one has to reduce a lattice basis of smaller dimension.

5 New Attack on Common Prime RSA

In this section, we explain how a small root of a polynomial $f(x, y, z) = a_0 + a_1x + a_2x^2 + a_3y + a_4z + a_5xy + a_6xz + a_7yz$ results in a new attack on a variant of RSA called Common Prime RSA. We show the following result.

Theorem 2 (Common Prime RSA)

Under Assumption 1, for every $\epsilon > 0$, there exists n_0 such that for every $n > n_0$, the following holds: Let $N = pq$ be an n -bit RSA modulus, and p, q primes of bitsize $\frac{n}{2}$ such that $p - 1 = 2ga$ and $q - 1 = 2gb$, for some prime g of bitsize γn , with $0 < \gamma < \frac{1}{2}$. Let $ed \equiv 1 \pmod{2gab}$, with $\text{bitsize}(e) = (1 - \gamma)n$ and $\text{bitsize}(d) = \delta n$, with $0 < \delta < (1 - \gamma)n$. Then d can be found in time polynomial in $\log N$ provided that

$$\delta < \frac{1}{4}(4 + 4\gamma - \sqrt{13 + 20\gamma + 4\gamma^2}) - \epsilon.$$

5.1 Common Prime RSA

In Section 4, we mentioned that a small d is unsafe in Wiener’s attack [19]. Therefore, RSA-CRT is often used when efficient decryption is needed. However, there is also a possibility to choose $d < N^{\frac{1}{4}}$ in RSA while avoiding Wiener’s attack. There is a variant of RSA where Wiener’s attack works less well, as was already shown by Wiener, namely when $\text{gcd}(p - 1, q - 1)$ has a large prime factor. Lim and Lee used this fact in a proposal [12], which was attacked a few years later by McKee and Pinch [15]. Recently Hinek [9] revisited this variant, calling it Common Prime RSA, and investigated its potential and its weaknesses.

In Common Prime RSA, we have $N = pq$ for primes p and q such that $p = 2ga + 1$ and $q = 2gb + 1$, for g a large prime, and a, b coprime integers. The exponents e and d are mutually inverse modulo $\text{lcm}(p - 1, q - 1) = 2gab$:

$$ed = 1 + k \cdot 2gab, \text{ with } 0 < e, d < 2gab.$$

The goal is to safely choose an exponent $d < N^{\frac{1}{4}}$, which enables a fast RSA decryption process. We set $g = N^\gamma$ and $d = N^\delta$ for some $0 \leq \gamma < \frac{1}{2}, 0 < \delta < 1 - \gamma$. Then, e is of size $N^{1-\gamma}$, k is of size N^δ , and a and b are both of size $N^{\frac{1}{2}-\gamma}$.

A large number of security issues were addressed in [9]. After excluding all parameter choices of Common Prime RSA that should be considered unsafe by the known attacks, Hinek concludes that there are still plenty of safe choices for $d = N^\delta$ with $\delta < \frac{1}{4}$ (see Fig. 1).

5.2 Description of the New Attack

An improved attack can be obtained by treating the equation in Hinek’s second lattice attack in a different way. In his attack, Hinek starts by multiplying the following two equations:

$$ed = 1 + k(p - 1)b, \quad ed = 1 + k(q - 1)a.$$

This can be written as $e^2d^2 + ed(ka + kb - 2) - (N - 1)k^2ab - (ka + kb - 1) = 0$. Next, he uses the fact that the polynomial $f(x, y, z, u) = e^2x + ey - (N - 1)z - u$ has a small root $(d^2, d(k(a + b - 2)), k^2ab, (ka + kb - 1))$. This leads to the bound $\delta < \frac{2}{5}\gamma$, for which the secret information can be revealed.

Now let us take another look at the equation

$$e^2d^2 + ed(ka + kb - 2) - (ka + kb - 1) - (N - 1)k^2ab = 0,$$

in which the unknowns are d, k, a and b . We can extract from this equation that the polynomial $f(x, y, z) = e^2x^2 + ex(y + z - 2) - (y + z - 1) - (N - 1)yz$ has a small root (d, ka, kb) with $X = N^\delta, Y = N^{\delta+\frac{1}{2}-\gamma}, Z = N^{\delta+\frac{1}{2}-\gamma}$. Moreover, $W = N^{2+2\delta-2\gamma}$.

Substituting these in the asymptotical bound $X^{7+9\tau+3\tau^2}(YZ)^{5+\frac{9}{2}\tau} < W^{3+3\tau}$ from Section 3 yields

$$3\delta\tau^2 + 3(4\delta - \frac{1}{2} - \gamma)\tau + (11\delta - 1 - 4\gamma) < 0.$$

For the optimal $\tau = \frac{\frac{1}{2}+\gamma-4\delta}{2\delta}$, this reduces to $\delta < \frac{1}{4}(4 + 4\gamma - \sqrt{13 + 20\gamma + 4\gamma^2})$.

Fig. 1 shows the new attack region as well as the known attacks, for any size of modulus N . Combinations of d and g that should be considered unsafe by the new attack are in the dark shaded area, whereas the lighter shaded area was already unsafe by the known attacks. It can be seen that the number of 'safe' combinations $\{d, g\}$ with $d < N^{\frac{1}{4}}$ has significantly decreased.

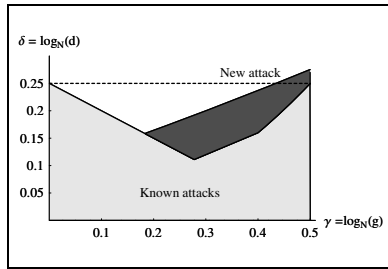


Fig. 1. New attack region

We note that for 'small' N (such as the regular 1024 bits), other attacks such as factoring attacks may apply, see [9]. Also, depending on the size of N , the attacks in the figure could be extended by an additive exhaustive search.

We performed experiments to check the validity of Assumption 1 and to demonstrate the practicality of our attack. We have implemented the new attack for the parameter setting $m = 2, t = 0$ (without the possible additional exhaustive search), to give an impression on what a realistic bound is for the smallest lattice possible. Of course, extending to $m = 3, m = 4$, etc. and using x -shifts will give results closer to the theoretical attack bound $\delta < \frac{1}{4}(4 + 4\gamma - \sqrt{13 + 20\gamma + 4\gamma^2})$, but will also result in a longer time needed for the lattice basis reduction. For $m = 2, t = 0$ the reduction time (the longest part of the attack) is about one minute.

The following table summarizes the experimental results performed for $m = 2, t = 0$, and $\log_2(N) = 1024$. As one can see, the results are already outside the asymptotical range of the two other lattice attacks described in [9].

γ	maximal δ (asymptotic) new attack	obtained δ ($m = 2, t = 0$) new attack	maximal δ (asymptotic) known attacks
0.10	0.130	0.07	0.20
0.20	0.164	0.10	0.15
0.30	0.200	0.13 (*)	0.12
0.40	0.237	0.17 (*)	0.16
0.50	0.275	0.2	0.25

The resultant heuristic of Assumption 1 worked perfectly in most cases. However, in the rare situation that both δ and γ were very small (e.g. $\gamma = 0.1$ and $\delta = 0.05$), we encountered cases where some of the polynomials h_i were algebraically dependent. In these cases, we could still recover the secret information in two different ways. One way was to use combinations of h_1 and the somewhat 'larger' h_i for $i > 2$, instead of only h_1 and h_2 . The other way was by examining the cause of the zero resultant. In essence, $\text{Res}_y(\text{Res}_x(h_1, f), \text{Res}_x(h_2, f)) = 0$ because $\text{Res}_x(h_1, f)$ and $\text{Res}_x(h_2, f)$ have a common polynomial factor, whose coefficients immediately reveal the secrets.

Acknowledgements. We thank Benne de Weger, Arjen Lenstra, Jason Hinek, and the anonymous reviewers for their helpful comments.

References

1. D. Boneh, G. Durfee: Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$, IEEE Transactions on Information Theory **46** [2000], 1339–1349.
2. J. Blömer, A. May: New Partial Key Exposure Attacks on RSA, Proceedings of CRYPTO 2003, LNCS **2729** [2003], 27–43.
3. J. Blömer, A. May: A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers, Proceedings of EUROCRYPT 2005, LNCS **3494** [2005], 251–267.
4. D. Coppersmith: Finding a Small Root of a Univariate Modular Equation, Proceedings of EUROCRYPT 1996, LNCS **1070** [1996], 155–165.
5. D. Coppersmith: Finding a Small Root of a Bivariate Integer Equation, Proceedings of EUROCRYPT 1996, LNCS **1070** [1996], 178–189.
6. D. Coppersmith: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities, Journal of Cryptology **10** [1997], 233–260.
7. J.-S. Coron: Finding Small Roots of Bivariate Integer Equations Revisited, Proceedings of EUROCRYPT 2004, LNCS **3027** [2004], 492–505.
8. M. Ernst, E. Jochemsz, A. May, B. de Weger: Partial Key Exposure Attacks on RSA up to Full Size Exponents, Proceedings of EUROCRYPT 2005, LNCS **3494** [2005], 371–386.
9. M.J. Hinek: Another Look at Small RSA Exponents, Proceedings of CT-RSA 2006, LNCS **3860** [2006], 82–98.
10. M.J. Hinek, D.R. Stinson: An Inequality About Factors of Multivariate Polynomials" [2006], <http://www.cacr.math.uwaterloo.ca/techreports/2006/cacr2006-15.pdf>.

11. N. Howgrave-Graham: Finding Small Roots of Univariate Modular Equations Revisited, *Cryptography and Coding*, LNCS **1355** [1997], 131–142.
12. C.H. Lim, P.J. Lee: Security and performance of server-aided RSA computation protocols, *Proceedings of CRYPTO 1995*, LNCS **963** [1995], 70–83.
13. A. Lenstra, H. Lenstra, Jr., L. Lovász: Factoring Polynomials with Rational Coefficients, *Mathematische Ann.* **261** [1982], 513–534.
14. A. May: New RSA Vulnerabilities Using Lattice Reduction Methods, PhD Thesis, University of Paderborn [2003].
15. J. McKee, R. Pinch: Further attacks on server-aided RSA cryptosystems [1998], <http://citeseer.ist.psu.edu/388295.html>.
16. P. Nguyen, D. Stehlé: Floating-Point LLL Revisited, *Proceedings of EUROCRYPT 2005*, LNCS **3494** [2005], 215–233.
17. G. Qiao, K.-Y. Lam: RSA Signature Algorithm for Microcontroller Implementation, *Proceedings of CARDIS 1998*, LNCS **1820** [2000], 353–356.
18. V. Shoup: NTL: A Library for doing Number Theory, online available at <http://www.shoup.net/ntl/index.html>.
19. M. Wiener: Cryptanalysis of Short RSA Secret Exponents, *IEEE Transactions on Information Theory* **36** [1990], 553–558.

A Small Modular Roots, Known Results

In this appendix, we give the known results for finding small modular roots [1,2,6] that can also be obtained by following the new strategy. Due to limited space, we only give the definitions of M_k that reproduce the known bounds. In all cases where the extended strategy is used, we use the notation $t = \tau m$ for some $\tau > 0$ that can be optimized later.

Boneh/Durfee [1]: $f_N(x_1, x_2) = a_0 + a_1x_1 + a_2x_1x_2$

The bound $X_1^{2+3\tau}X_2^{1+3\tau+3\tau^2} < N^{1+3\tau}$ can be found with the extended strategy using $x_1^{i_1}x_2^{i_2} \in M_k \Leftrightarrow i_1 = k, \dots, m; i_2 = k, \dots, i_1 + t$

Blömer/May [2]: $f_N(x_1, x_2, x_3) = a_0 + a_1x_1 + a_2x_2 + a_3x_2x_3$

The bound $X_1^{1+4\tau}X_2^{2+4\tau}X_3^{1+4\tau+6\tau^2} < N^{1+4\tau}$ can be found with the extended strategy, with $x_1^{i_1}x_2^{i_2}x_3^{i_3} \in M_k \Leftrightarrow i_1 = k, \dots, m; i_2 = 0, \dots, m - i_1; i_3 = 0, \dots, i_2 + t$.

Generalized Rectangle (generalization of a bound of Coppersmith[6]):

$f_N(x_1, \dots, x_n)$ is a polynomial such that the degree of x_i is $\lambda_i D$.

The bound $X_1^{\lambda_1} \dots X_n^{\lambda_n} < N^{\frac{2}{(n+1)D}}$ can be obtained with the basic strategy using $x_1^{i_1} \dots x_n^{i_n} \in M_k \Leftrightarrow i_j = \lambda_j Dk, \dots, \lambda_j Dm$ (for $j = 1, \dots, n$)

Generalized Lower Triangle (generalization of a bound of Coppersmith[6]):

$f_N(x_1, \dots, x_n)$ is a polynomial with monomials $x_1^{i_1} \dots x_n^{i_n}$ for $i_1 = 0, \dots, \lambda_1 D$, $i_2 = 0, \dots, \lambda_2 D - \frac{\lambda_2}{\lambda_1} i_1, \dots, i_n = 0, \dots, \leq \lambda_n D - \sum_{r=1}^{n-1} \frac{\lambda_n D}{\lambda_r} i_r$.

The bound $X_1^{\lambda_1} \dots X_n^{\lambda_n} < N^{\frac{1}{D}}$ can be obtained with the basic strategy, with $x_1^{i_1} \dots x_n^{i_n} \in M_k \Leftrightarrow i_1 = \lambda_1 Dk, \dots, \lambda_1 Dm; i_j = 0, \dots, \lambda_j Dm - \sum_{r=1}^{j-1} \frac{\lambda_j}{\lambda_r} i_r$ (for $j = 2, \dots, n$).

B Small Integer Roots, Known Results

In this appendix, we give the known results for finding small integer roots [3,8,6] that can also be obtained with the basic or extended strategy. Due to limited space, we only give the definitions of S and M that reproduce the known bounds. In all cases where the extended strategy is used, we use the notation $t = \tau m$ for some $\tau > 0$ that can be optimized later.

Blömer/May, Upper Triangle [3]:

$f(x_1, x_2)$ is a polynomial with monomials $x_1^{i_1} x_2^{i_2}$ for $i_1 = 0 \dots D, i_2 = 0 \dots \lambda i_2$.

The bound $X_1^{(\lambda+\tau)^2} X_2^{2(\lambda+\tau)} < W^{\frac{1}{D}(\lambda+2\tau)}$ can be obtained with the extended strategy, with $x_1^{i_1} \dots x_n^{i_n} \in S \Leftrightarrow i_2 = 0, \dots, D(m-1); i_1 = 0, \dots, \lambda i_2 + t$, and $x_1^{i_1} \dots x_n^{i_n} \in M \Leftrightarrow i_2 = 0, \dots, Dm; i_1 = 0, \dots, \lambda i_2 + t$.

Blömer/May, Extended Rectangle [3]:

$f(x_1, x_2)$, with monomials $x_1^{i_1} x_2^{i_2}$ for $i_2 = 0, \dots, D, i_1 = 0, \dots, \gamma D + \lambda(D - i_2)$, e.g. $f(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3 + a_4 x_2 + a_5 x_1 x_2$ (where $D = 1, \gamma = 1, \lambda = 2$). The bound $X_1^{\lambda^2+3\gamma\lambda+2\tau\lambda+4\tau\gamma+\tau^2+3\gamma^2} X_2^{\lambda+3\gamma+2\tau} < W^{\frac{1}{D}(\lambda+2\gamma+2\tau)}$ can be obtained with the extended strategy, using $x_1^{i_1} x_2^{i_2} \in S \Leftrightarrow i_2 = 0, \dots, D(m-1); i_1 = 0, \dots, \gamma D(m-1) + \lambda(D(m-1) - i_2) + t$, and $x_1^{i_1} x_2^{i_2} \in M \Leftrightarrow i_2 = 0, \dots, Dm; i_1 = 0, \dots, \gamma Dm + \lambda(Dm - i_2) + t$.

Ernst et al. 1 [8]: $f(x_1, x_2, x_3) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_2 x_3$.

The bound $X_1^{1+3\tau} X_2^{2+3\tau} X_3^{3+3\tau+3\tau^2} < W^{1+3\tau}$ can be found with the extended strategy, with $x_1^{i_1} x_2^{i_2} x_3^{i_3} \in S \Leftrightarrow i_1 = 0, \dots, m-1; i_2 = 0, \dots, m-1-i_1; i_3 = 0, \dots, i_2+t$, and $x_1^{i_1} x_2^{i_2} x_3^{i_3} \in M \Leftrightarrow i_1 = 0, \dots, m; i_2 = 0, \dots, m-i_1; i_3 = 0, \dots, i_2+t$.

Ernst et al. 2 [8]: $f(x_1, x_2, x_3) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_2 x_3$.

The bound $X_1^{2+3\tau} X_2^{3+3\tau} X_3^{3+6\tau+3\tau^2} < W^{2+3\tau}$ can be found with the extended strategy, using $x_1^{i_1} x_2^{i_2} x_3^{i_3} \in S \Leftrightarrow i_1 = 0, \dots, m-1; i_2 = 0, \dots, m-1-i_1+t; i_3 = 0, \dots, m-1-i_1$, and $x_1^{i_1} x_2^{i_2} x_3^{i_3} \in M \Leftrightarrow i_1 = 0, \dots, m; i_2 = 0, \dots, m-i_1+t; i_3 = 0, \dots, m-i_1$.

Generalized Rectangle (generalization of a bound of Coppersmith [6]):

$f(x_1, \dots, x_n)$ is a polynomial where the degree of x_i is $\lambda_i D$.

The bound $X_1^{\lambda_1} \dots X_n^{\lambda_n} < W^{\frac{2}{(n+1)D}}$ can be found with the basic strategy, with $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in S \Leftrightarrow i_j = 0, \dots, \lambda_j D(m-1)$, and $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in M \Leftrightarrow i_j = 0, \dots, \lambda_j Dm$ (for $j = 1, \dots, n$).

Generalized Lower Triangle (generalization of a bound of Coppersmith [6]):

$f(x_1, \dots, x_n)$ is a polynomial monomial are $x_1^{i_1} \dots x_n^{i_n}$ for $0 \leq i_1 \leq \lambda_1 D, 0 \leq i_2 \leq \lambda_2 D - \frac{\lambda_2}{\lambda_1} i_1, \dots, 0 \leq i_n \leq \lambda_n D - \sum_{r=1}^{n-1} \frac{\lambda_n}{\lambda_r} i_r$.

The bound $X_1^{\lambda_1} \dots X_n^{\lambda_n} < W^{\frac{1}{D}}$ can be found with the basic strategy, with $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in S \Leftrightarrow i_j = 0, \dots, \lambda_j D(m-1) - \sum_{r=1}^{j-1} \frac{\lambda_j}{\lambda_r} i_r$, and $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \in M \Leftrightarrow i_j = 0, \dots, \lambda_j Dm - \sum_{r=1}^{j-1} \frac{\lambda_j}{\lambda_r} i_r$ (for $j = 1, \dots, n$).

Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding

Donghoon Chang¹, Sangjin Lee¹, Mridul Nandi², and Moti Yung³

¹ Center for Information Security Technologies(CIST), Korea University, Seoul, Korea
{dhchang, sangjin}@cist.korea.ac.kr

² David R. Cheriton School of Computer Science, University of Waterloo, Canada
m2nandi@cs.uwaterloo.ca

³ RSA Laboratories and Department of Computer Science, Columbia University,
New York, USA
moti@cs.columbia.edu

Abstract. Understanding what construction strategy has a chance to be a good hash function is extremely important nowadays. In TCC'04, Maurer *et al.* [13] introduced the notion of indifferentiability as a generalization of the concept of the indistinguishability of two systems. In Crypto'2005, Coron *et al.* [5] suggested to employ indifferentiability in generic analysis of hash functions and started by suggesting four constructions which enable eliminating all possible generic attacks against iterative hash functions. In this paper we continue this initial suggestion and we give a formal proof of indifferentiability and indifferentiable attack for prefix-free MD hash functions (for single block length (SBL) hash and also some double block length (DBL) constructions) in the random oracle model and in the ideal cipher model. In particular, we observe that there are sixteen PGV hash functions (with prefix-free padding) which are indifferentiable from random oracle model in the ideal cipher model.

1 Introduction

The notion of indifferentiability was first introduced by Maurer *et al.* [13] and is a stronger notion than indistinguishability. For example, assume a cryptosystem $\mathcal{P}(\mathcal{G})$ based on a random oracle \mathcal{G} is secure. Now, to prove the security of $\mathcal{P}(H^{\mathcal{F}})$ based on Merkle-Damgard (MD) hash function H where the underlying compression function is a random oracle, we need to prove something different than indistinguishability. In fact, we need to prove that $H^{\mathcal{F}}$ is indifferentiable (as was introduced in [13]) from a random oracle. Informally, $H^{\mathcal{F}}$ is indifferentiable from random oracle if there is no efficient attacker (or distinguisher) which can distinguish \mathcal{F} and the hash function based on it from a random oracle R and an efficient simulator of \mathcal{F} . Here R is a random oracle with (finite) domain and range same as that of H . In case of Indistinguishability, the distinguisher only needs to tell apart H from \mathcal{G} without any help of oracle \mathcal{F} . Thus, the notion of indifferentiability is important when we consider attacks on a cryptosystem based on some ideal primitive where the attacker has some access on the computation of the primitive. In the case of hash function $H^{\mathcal{F}}$, the attacker can also

compute \mathcal{F} as it is a random oracle which can be computed publicly. So this new notion is important for stronger attackers. If the attacker does not have that access (to the random oracle) then merely indistinguishability will suffice to preserve the security of the cryptosystem.

Recently, Coron *et al.* [5] suggested to employ the notion for analysis of hash functions and they proved that the classical MD iteration is not indistinguishable with random oracle when the underlying compression function is random oracle. They have also stated indistinguishability for prefix-free MD hash functions or some other definition of hash functions like HMAC, NMAC, chop-MD hash function. They also have stated indistinguishability for Davis-Meyer construction (which is one of the classical PGV construction [17]) in the ideal cipher model.

Our Results: In this paper we extend the use of indistinguishability in analyzing hash functions, and we present a proof methodology for determining indistinguishability. We discuss indistinguishability of several known hash constructions with the random oracle model including the prefix free MD hash function. We consider all collision secure PGV hash functions in the ideal cipher model [2] (there are twenty such hash functions). It is easy to check that under ideal cipher model the underlying compression function is not indistinguishable with random oracle. So we can not use the indistinguishability result directly for prefix-free MD hash function (where we need the underlying compression function as a random oracle). But we will show that out of twenty, sixteen hash functions with prefix free padding are indistinguishable from random oracle. We also prove the indistinguishability of some known Double length hash functions in the random oracle model for the underlying single length compression function. Finally, we will also show several differentiability attacks on block-cipher based on double length hash function namely, PBGV, LOKI-DBH, MDC2 etc.

Organization: The organization of this paper is as follows. In section 2, we define notations and describe the security notion of indistinguishability with some mathematical background and notations which will help to prove the security later. In section 3, we provide formal proofs of prefix-free single length MD hash functions, PGV hash functions, and double length hash function. Then, in section 4, we show the differentiability of some SBL and DBL hash functions. Finally we conclude.

2 Preliminaries and Related Work

In this section, we briefly describe random oracle and ideal cipher model and we review how the adversary works in these model. Then some designs of hash functions are stated.

2.1 Ideal Model and Iterated Structure

Random Oracle Model: f is said to be a *random oracle* from X to Y if for each $x \in X$ the value of $f(x)$ is chosen randomly from Y . More precisely, $\Pr[f(x) = y \mid f(x_1) = y_1, f(x_2) = y_2, \dots, f(x_q) = y_q] = \frac{1}{M}$, where

$x \notin \{x_1, \dots, x_q\}$, $y, y_1, \dots, y_q \in Y$ and $|Y| = M$. There is an equivalent way to look a random function: Consider $\text{Map}(X \rightarrow Y)$, the set of all mappings from X to Y . f is said to be a random oracle if it is chosen uniformly from the set $\text{Map}(X \rightarrow Y)$. The adversary \mathcal{A} can only query f adaptively, say by inputting x_1, \dots, x_q , where q is the total number of queries. Let y_1, \dots, y_q be the responses of these queries, i.e., $f(x_1) = y_1, \dots, f(x_q) = y_q$. Since an adversary makes queries adaptively, the i^{th} query x_i only depends on previous query-responses (in short, q-r) $(x_1, y_1), \dots, (x_{i-1}, y_{i-1})$ and on the random coins selected by the adversary.

Ideal Cipher Model: Ideal cipher model is the one dating back to Shannon [19] and used, e.g., in [7,10,20]. Let $\text{Bloc}(\mathcal{K}, X) = \{E : \mathcal{K} \times X \rightarrow X; E(k, \cdot) \text{ is a permutation for each } k \in \mathcal{K}\}$. As above, a function E is chosen uniformly from the set $\text{Bloc}(\mathcal{K}, X)$. As $E(k, \cdot)$ (we also use the notation $E_k(\cdot)$) is a permutation, an adversary \mathcal{A} can have access to two oracles E and E^{-1} . Thus, the q-r's look like $(\sigma_1, k_1, x_1, y_1), \dots, (\sigma_q, k_q, x_q, y_q)$, where $\sigma_i = \pm 1$ and $E_{k_i}(x_i) = y_i, i \leq i \leq q$. If $\sigma_i = 1$ then adversary makes E query with input (k_i, x_i) and response is y_i and if $\sigma_i = -1$ then adversary makes E^{-1} query with input (k_i, y_i) and response is x_i . Now one can check that, for each k , $E_k(\cdot)$ behaves like a random permutation (i.e., $\Pr[E_k(x) = y \mid E_k(x_1) = y_1, \dots, E_k(x_q) = y_q] = \frac{1}{M-q}$, where $x \notin \{x_1, \dots, x_q\}, y \notin \{y_1, \dots, y_q\} \subseteq Y$ and $|Y| = M$) and for different choices of keys $k_1, \dots, k_l, E_{k_1}(\cdot), \dots, E_{k_l}(\cdot)$ are independently distributed. See [2] for more details and discussions about black-box models.

Iterated Hash Function: Now given a function $F : Y \times B \rightarrow Y$, one can define an iterated function $F^* : Y \times B^* \rightarrow Y$ as follows :

$$F^*(x, m_1, m_2, \dots, m_l) = F(\dots F(x, m_1), \dots, m_l), m_i \in B, x \in Y$$

where $B^* = \cup_{i \geq 0} B^i$. Let \mathcal{M} be a message space (finite) and $g : \mathcal{M} \rightarrow B^*$ be any function called a padding rule. Then the MD-Hash function based on a compression function F , a fixed initial value $\text{IV} \in Y$ and a padding rule $g(\cdot)$ is $\text{MD}_g^F(M) = F^*(\text{IV}, g(M))$. A padding rule is called a prefix-free if $M_1 \neq M_2 \Rightarrow g(M_1)$ is not a prefix of $g(M_2)$. Coron *et al.* [5] considered prefix-free MD iteration and suggested indifferentiability from random oracle model.

Given a compression function $F : Y \times B \rightarrow Y$, one can also define a wide compression function $W : Y' \times B' \rightarrow Y'$, where Y' is a bigger set than Y . For example, in case of a double length compression function $Y' = Y \times Y$. An example of a general class of double length compression functions due to Nandi [15] is as follows : $W(x_1, x_2, m) = F(x_1 \parallel x_2, m) \parallel F(p(x_1 \parallel x_2), m)$, where $x_1, x_2 \in Y, m \in B', F : Y \times (Y \times B') \rightarrow Y$ and p is a permutation on $Y \times Y$ so that it does not have any fixed point (y is called fixed point of p if $p(y) = y$).

2.2 Known Results on Indifferentiability

In this section we give a brief introduction of indifferentiability and state some known results on it.

Definition 1. [5] A Turing machine C with oracle access to an ideal primitive \mathcal{F} is said to be $(t_D, t_S, q, \varepsilon)$ indiffereniable from an ideal primitive \mathcal{G} if there exists a simulator S such that for any distinguisher D it holds that :

$$|\Pr[D^{C,\mathcal{F}} = 1] - \Pr[D^{\mathcal{G},S} = 1]| < \varepsilon$$

The simulator has oracle access to \mathcal{G} and runs in time at most t_S . The distinguisher runs in time at most t_D and makes at most q queries. Similarly, $C^{\mathcal{F}}$ is said to be (computationally) indiffereniable from \mathcal{G} if ε is a negligible function of the security parameter k (for polynomially bounded t_D and t_S).

In this paper, we will mainly consider $C = H^{\mathcal{F}}$, where H is MD (or prefix-free MD) hash function based on the random oracle model (or ideal cipher model) \mathcal{F} and \mathcal{G} is a random oracle with same domain and range as the hash function. In case of ideal cipher model the distinguisher can access both \mathcal{F} and \mathcal{F}^{-1} oracles and the simulator has to simulate both.

The following Theorem [13] due to Maurer *et al.* is related to this paper. We explain the theorem for random oracle model of hash functions. Suppose a hash function (in some design of iteration) H based on a random oracle (or an ideal cipher) \mathcal{F} is indiffereniable from a random oracle \mathcal{G} . Then a cryptosystem \mathcal{P} based on the random oracle \mathcal{G} is at least as secure as the cryptosystem \mathcal{P} based on the hash function H in the random oracle model (or an ideal cipher model) \mathcal{F} . Here, \mathcal{F} is the underlying compression function of H (or block-cipher in case of block cipher based hash function). The original theorem as stated below is a more general statement.

Theorem 1. [13] *Let \mathcal{P} be a cryptosystem with oracle access to an ideal primitive \mathcal{G} . Let H be an algorithm such that $H^{\mathcal{F}}$ is indiffereniable from \mathcal{G} . Then cryptosystem \mathcal{P} is at least as secure in the \mathcal{F} model with algorithm H as in the \mathcal{G} model.*

Coron *et al.* stated the indiffereniable of prefix free MD construction in random oracle (or in ideal cipher model in the case of block-cipher based construction). In [5] the following theorems are stated.

Theorem 2. [5] *The prefix-free MD construction is $(t_D, t_S, q, \varepsilon)$ -indiffereniable from a random oracle, in the random oracle model for the compression function, for any t_D , with $t_S = \ell \cdot O(q^2)$ and $\varepsilon = 2^{-n} \cdot \ell^2 \cdot O(q^2)$, where ℓ is the maximum length of a query made by the distinguisher D .*

Theorem 3. *The Davis-Meyer Hash function (based on the compression function $f(x, m) = E_m(x) \oplus x$ and a prefix free padding g) MD_g^f is $(t_D, t_S, q, \varepsilon)$ -indiffereniable from a random oracle, in the ideal cipher model, for any t_D , with $t_S = \ell \cdot O(q^2)$ and $\varepsilon = 2^{-n} \cdot \ell^2 \cdot O(q^2)$, where ℓ is the maximum length of a query made by the distinguisher D .*

2.3 Adversary in the Random Oracle Model

A binary relation \mathcal{R} on $(X \times B, X)$ is a subset of $X \times B \times X$. A relation is called *functional relation* (or *partial functional relation*) if for each $(x, m) \in X \times B$ there

exists at most one $y \in X$ such that $(x, m, y) \in \mathcal{R}$. Thus, a partial functional relation is uniquely characterized by a partial function $f : X \times B \rightarrow X$ (a partial function may have some points on domain where the functional value is not defined). Now given a relation \mathcal{R} on $(X \times B) \times X$, one can define a *functional closure* relation \mathcal{R}^* on $(X \times B^*) \times X$ which is a minimal relation containing \mathcal{R} such that following are true:

1. $(x_1, M_1, x_2), (x_2, M_2, x_3) \in \mathcal{R}^* \implies (x_1, M_1 \parallel M_2, x_3) \in \mathcal{R}^*$.
2. $(x_1, M_1 \parallel M_2, x_3), (x_1, M_1, x_2) \in \mathcal{R}^* \implies (x_2, M_2, x_3) \in \mathcal{R}^*$.

Thus, if \mathcal{R} corresponds to a partial function $f : X \times B \rightarrow X$, then \mathcal{R}^* corresponds to the partial function f^* which is obtained from the partial function f iteratively. Sometimes, we use a more appealing notation $x_1 \rightarrow_{M_1} x_2 \in \mathcal{R}$ (or $x_1 \rightarrow_{M_1} x_2$ when the relation is clear from the context) to denote that $(x_1, M_1, x_2) \in \mathcal{R}^*$. Thus, in terms of this notation, \mathcal{R}^* is the minimal relation containing \mathcal{R} with the following conditions:

1. If $x_1 \rightarrow_{M_1} x_2 \rightarrow_{M_2} x_3$, then $x_1 \rightarrow_{M_1 \parallel M_2} x_3$ (transitive property).
2. If $x_1 \rightarrow_{M_1} x_2$ and $x_1 \rightarrow_{M_1 \parallel M_2} x_3$, then $x_2 \rightarrow_{M_2} x_3$ (substitute property).

Let D be a distinguisher (or an adversary) in the indifferentiable attack. He has an access to two oracles \mathcal{O}_1 and \mathcal{O}_2 . In this scenario, either $(\mathcal{O}_1, \mathcal{O}_2) = (H, f)$ or $(\mathcal{O}_1, \mathcal{O}_2) = (\text{Rand}, S)$, where $H = \text{MD}_g^f$ (prefix free MD hash function with fixed initial value IV), S is any simulator, f and Rand are random oracles from $X \times B$ to X and from \mathcal{M} to X respectively. Distinguisher is making successive queries of \mathcal{O}_1 or \mathcal{O}_2 . Suppose the i^{th} query is an \mathcal{O}_1 query with the message $M \in \mathcal{M}$ and the response of the query is h (say), then we write $r_i = \text{IV} \rightarrow_{g(M)} h$. Otherwise, $r_i = h_1 \rightarrow_m h_2$ for \mathcal{O}_2 query (h_1, m) with response h_2 . Let $\mathcal{R}_i = \{r_1, \dots, r_i\}$ be the relation characterizing the query-response after the i^{th} query and \mathcal{R}_i^* be the functional closure of \mathcal{R}_i characterizing the view of the distinguisher after i^{th} query. Thus, $\mathcal{Q} = (\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_q)$ be the complete query-response tuple and $\mathcal{V} = (\mathcal{R}_1^*, \mathcal{R}_2^*, \dots, \mathcal{R}_q^*)$ be the complete view of the distinguisher D , where q is the total number of queries. Now we define some terminology which will be useful in this context.

1. Define *support* of a relation \mathcal{R}_i by a subset of X , $\text{Supp}(\mathcal{R}_i) = \{h : h \rightarrow_m h_1 \in \mathcal{R}_i\} \cup \{h : h_1 \rightarrow_m h \in \mathcal{R}_i\} \cup \{\text{IV}\}$. Note that, $\text{Supp}(\mathcal{R}_i) = \text{Supp}(\mathcal{R}_i^*)$.
2. We say, r_i is a *trivial query* if $r_i \in \mathcal{R}_{i-1}^*$. Since g is a prefix-free padding, r_i can be trivial query only if any one of the following holds :
 - (a) $r_i = \text{IV} \rightarrow_{g(M)} h_\ell$, where $\text{IV} = h_0 \rightarrow_{m_1} h_1 \rightarrow_{m_2} \dots h_{\ell-1} \rightarrow_{m_\ell} h_\ell \in \mathcal{R}_{i-1}^*$ and $g(M) = m_1 \parallel \dots \parallel m_\ell$.
 - (b) $r_i = h_{\ell-1} \rightarrow_{m_\ell} h_\ell$, where $\text{IV} = h_0 \rightarrow_{m_1} h_1 \rightarrow_{m_2} \dots h_{\ell-1}$, $\text{IV} \rightarrow_{g(M)} h_\ell \in \mathcal{R}_{i-1}^*$ and $g(M) = m_1 \parallel \dots \parallel m_\ell$.
 - (c) r_i is a repetition query i.e. $r_i = r_j$ for some $j < i$. For simplicity, we can assume that there is no repetition query as distinguisher's point of view it does not help anything.
3. We say \mathcal{V} is not *collision free* (or in short $\neg \text{CF}$) if for some i , $r_i = h \rightarrow_M h'$ is non trivial and $h' \in \text{Supp}(\mathcal{R}_{i-1}) \cup \{h\}$.

3 Security Analysis

In this section, we explain how to obtain a formal proof of indistinguishability of prefix-free single length or double length or block-cipher based MD hash functions. Let E be an event which is only a function of the view of the distinguisher. In this case we consider complement of the collision-free event ($\neg CF$). Thus, there are events E_1 and E_2 for E when D interact with (H, f) and $(Rand, S)$, respectively. If this event is defined carefully so that

1. (H, f) and $(Rand, S)$ are identically distributed conditioned on the past view of the distinguisher and E does not occur, and
2. if $\Pr[E_1], \Pr[E_2] \leq \max$, where \max is some negligible function.

Because of item 1, $\Pr[D^{H,f} \rightarrow 1 \mid \neg E_1] = \Pr[D^{R,S} \rightarrow 1 \mid \neg E_2]$. Then, one can show the indistinguishability of H with the random oracle model. More precisely,

$$\begin{aligned}
 Adv(D) &= | \Pr[D^{H,f} \rightarrow 1] - \Pr[D^{R,S} \rightarrow 1] | \\
 &= | \Pr[D^{H,f} \rightarrow 1 \mid E_1] \times \Pr[E_1] + \Pr[D^{H,f} \rightarrow 1 \mid \neg E_1] \times \Pr[\neg E_1] \\
 &\quad - \Pr[D^{R,S} \rightarrow 1 \mid E_2] \times \Pr[E_2] - \Pr[D^{R,S} \rightarrow 1 \mid \neg E_2] \times \Pr[\neg E_2] | \\
 &\leq \max \times | \Pr[D^{H,f} \rightarrow 1 \mid E_1] - \Pr[D^{R,S} \rightarrow 1 \mid E_2] | \\
 &\quad + \Pr[D^{H,f} \rightarrow 1 \mid \neg E_1] \times | \Pr[\neg E_1] - \Pr[\neg E_2] | \quad \dots\dots (1) \\
 &= \max \times | \Pr[D^{H,f} \rightarrow 1 \mid E_1] - \Pr[D^{R,S} \rightarrow 1 \mid E_2] | \\
 &\quad + \Pr[D^{H,f} \rightarrow 1 \mid \neg E_1] \times | \Pr[E_1] - \Pr[E_2] | \quad \dots\dots (2) \\
 &\leq \max \times | \Pr[D^{H,f} \rightarrow 1 \mid E_1] - \Pr[D^{R,S} \rightarrow 1 \mid E_2] | \\
 &\quad + \max \times \Pr[D^{H,f} \rightarrow 1 \mid \neg E_1] \\
 &\leq 2 \times \max
 \end{aligned}$$

In (1), $\Pr[D^{H,f} \rightarrow 1 \mid \neg E_1] = \Pr[D^{R,S} \rightarrow 1 \mid \neg E_2]$ and in (2), $\Pr[\neg E_2] - \Pr[\neg E_1] = \Pr[E_1] - \Pr[E_2]$. Thus we have,

$$Adv(D) \leq 2 \times \max\{\Pr[E_1], \Pr[E_2]\} \quad \dots\dots (3)$$

Similarly, if H is based on the block cipher E , we have three set of oracles (H^E, E, E^{-1}) or $(Rand, S, S^{-1})$. Then we can proceed as like above.

3.1 Indistinguishability of Prefix Free Single Length MD Hash Functions

Now we define a simulator S which simulates f so that no distinguisher can distinguish (R, S) with (H, f) , where R and f are assumed to be random oracles and H is the prefix-free hash function based on f .

Simulator: The simulator keeps the relations $(\mathcal{R}_1, \dots, \mathcal{R}_{i-1})$. Initially, $\mathcal{R}_0 = \emptyset$. On the i^{th} query (h_i, x_i) , the response of S is as follow

1. If $\exists IV \rightarrow_N h_i \in \mathcal{R}_{i-1}, g(M) = N \parallel x_i$, then run $\text{Rand}(M)$ and obtain the response h^* . $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_i \rightarrow_{x_i} h^*\}$ and return h^* . For more than one choices of M , return a random string h^* (this will never happen if $(\mathcal{R}_1, \dots, \mathcal{R}_q)$ is collision-free).
2. Else return a random string h^* and $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_i \rightarrow_{x_i} h^*\}$.

If distinguisher is making at most q queries then one can design the above simulator so that it runs in time $O(\ell q)$. In the worst case, simulator has to back track to initial value to check whether condition (1) is satisfied or not and this is needed at most $O(\ell q)$ time. Note that in [5] time complexity for simulator is $O(\ell q^2)$.

Distribution of oracles: Here, we study the conditional distribution of all oracles given the past view of the distinguisher and the collision-freeness of the view.

Let \mathcal{Q}_i be the set of all query-response after i queries. Let CF_1 and CF_2 denote that the complete view \mathcal{V} is collision free (CF) in case of (H, f) and (Rand, S) queries, respectively. Given $\mathcal{Q}_{i-1} \wedge \text{CF}$, the i^{th} query r_i is a trivial query in (H, f) if and only if so is in (Rand, S) and the response of the trivial query is uniquely determined by the previous view. So, output of H or S is same as output of Rand or S respectively. So assume that r_i is not a trivial query.

Lemma 1. *Given $\mathcal{Q}_{i-1} \wedge \text{CF}$, the conditional distribution of H, f, Rand and S on i^{th} query (h_i, x_i) is uniformly distributed on the set $X \setminus (\text{Supp}(\mathcal{R}_{i-1}) \cup \{h_i\})$ provided it is not a trivial query ($h_i = IV$ for \mathcal{O}_1 oracle query).*

Proof. In case of Rand and S , as CF_2 is not true the output is drawn randomly outside the set $\text{Supp}(\mathcal{R}_{i-1}) \cup \{h_i\}$. In case of Rand query M , since r_i is a nontrivial query, $\text{Rand}(M)$ hash has not been queried before even by the simulator. So, condition on CF_2 the distribution of $\text{Rand}(M)$ is uniformly distributed on the set $X \setminus (\text{Supp}(\mathcal{R}_{i-1}) \cup \{IV\})$. In case of S query (h_i, x_i) query, the output is not random only if it is trivial query (where the case (1) of the simulator occurs and for the corresponding message M $\text{Rand}(M)$ has been queried before by the distinguisher). So it is true for both Rand and S . Now we will prove it for H .

Let $S_i = \text{Supp}(\mathcal{R}_{i-1}) \cup \{h_i\}$. If we can prove that for all $a \neq a' \notin S_i$, $\Pr[H(M) = a \mid \mathcal{Q}_{i-1} \wedge \text{CF}_1] = \Pr[H(M) = a' \mid \mathcal{Q}_{i-1} \wedge \text{CF}_1]$ then we are done since for all other choices of a the probability is zero because of condition of CF_1 . Given a and a' , Let $A = \{f : X \times B \rightarrow X : H^f(M) = a \wedge f \text{ satisfies } \mathcal{Q}_{i-1}\}$. Similarly define A' for a' . Now one can define a bijection ϕ between A and A' in the following way.

1. If $f \in A$ then $\phi(f)(h, x) = f(h, x)$ if $\{f(h, x), h\} \cap \{a, a'\} = \emptyset$
2. $\phi(f)(a, x) = f(a', x)$ if $f(a', x) \notin \{a, a'\}$. Similarly, $\phi(f)(a', x) = f(a, x)$ if $f(a, x) \notin \{a, a'\}$.

3. If $h \notin \{a, a'\}$ but $f(h, x) = a$ then $\phi(f)(h, x) = a'$. Similarly, $f(h, x) = a'$ then $\phi(f)(h, x) = a$.
4. There are four other possibilities i.e.
 - (a) if $f(a, x) = a$ then $\phi(f)(a', x) = a'$.
 - (b) if $f(a, x) = a'$ then $\phi(f)(a', x) = a$.
 - (c) if $f(a', x) = a$ then $\phi(f)(a, x) = a'$.
 - (d) if $f(a', x) = a'$ then $\phi(f)(a, x) = a$.

Now it is easy to check that $\phi(f)$ is well defined and it belongs to A' . Here, we mainly interchange the role of a and a' in all possible cases of input and output keeping other values the same. Thus, given $H^f(M) = a$, we should have $H^{\phi(f)}(M) = a'$ keeping all other equalities fixed (in \mathcal{Q}_{i-1}). Now it is also easy to check that this is a bijection as we can define the inverse function similarly. Thus, $|A| = |A'|$ and hence the probabilities are equal. We can prove similarly for the distribution of f . So we skip the proof of this. ■

Now we bound the probability of collision events for both cases.

Lemma 2. $\Pr[\neg CF_1] = O(\frac{l^2 q^2}{2^n})$ and $\Pr[\neg CF_2] = O(\frac{q^2}{2^n})$, where l is the maximum number of blocks in H -query and $|X| = 2^n$.

Proof. We first assume that there is no trivial query. If it is there, then we have less probability as it does not help in collision. Now we compute the probability where all outputs (including the intermediate hash values for different messages) and inputs of f are distinct. Now any choices of input-outputs satisfying the above give all different inputs to f . Thus, the probability of any such choice is $1/2^{nq'}$, where q' is the total number of inputs of f . Number of choices of above tuples is at least $(|X| - 1)(|X| - 3) \cdots (|X| - 2q' + 1)$. Thus, $\Pr[CF_1] = (|X| - 1)(|X| - 3) \cdots (|X| - 2q' + 1)/2^{nq'} = 1 - O(\frac{l^2 q^2}{2^n})$. In case of $\Pr[CF_2]$, the probability is $O(\frac{q^2}{2^n})$ as output of simulator and Rand is random except for nontrivial query. As nontrivial can not make collision we have the above collision probability. ■

Combining the lemmas and Equation (3) we obtain the following main theorem of this section.

Theorem 4. *Prefix-free single length MD hash functions in a fixed-size random oracle model is (t_D, t_S, q, ϵ) -indifferentiable from a random oracle, for any t_D , with $t_S = l \cdot \mathcal{O}(q)$ and $\epsilon = 2^{-n+1} \cdot l^2 \cdot \mathcal{O}(q^2)$, where l is the maximum length of a query made by the distinguisher D .*

3.2 Indifferentiability of Prefix Free PGV Hash Functions

Now we consider all collision secure PGV hash functions. We will show, in the prefix-free mode, that sixteen (indexed by 1 ~ 16 in table 1 of Appendix A) out of twenty are also indifferentiable with random oracle. Others (indexed by 17 ~ 20 in table 1 of Appendix A) are not indifferentiable from random oracle.

It is easy to check that any PGV compression functions are not indifferentiable with random oracle.

Thus, we can not apply the previous theorem directly. First we consider the previous example $f(h_{i-1}, m_i) = E_{m_i}(h_{i-1}) \oplus h_{i-1}$. Coron *et al.* also considered this example and stated indifferentiability in [5]. We will define a simulator which simulates both E and E^{-1} . On query $(1, \cdot, \cdot)$ it simulates E and on query $(-1, \cdot, \cdot)$ it simulates E^{-1} .

Simulator. Like the previous simulator, it also keeps the relations $(\mathcal{R}_1, \dots, \mathcal{R}_{i-1})$. Initially, $\mathcal{R}_0 = \emptyset$. Let $\{P_x\}_{x \in X}$ be a family of random permutation. Now the response of S is as follow:

1. On query $(1, h_i, x_i)$,
 - (a) If $\mathbb{IV} \rightarrow_N h_i$ and $g(M) = N \parallel x_i$ then run $\text{Rand}(M)$ and obtain the response h^* . Return $h^* \oplus h_i$ and $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_i \rightarrow_{x_i} h^*\}$ (otherwise behave randomly and similar to previous simulator this does not occur if collision-free occurs).
 - (b) Else return $P_{x_i}(h_i) = h^*$, $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_i \rightarrow_{x_i} h^* \oplus h_i\}$.
2. On query $(-1, y_i, x_i)$,
 - (a) For each $\mathbb{IV} \rightarrow_N h$ such that $g(M) = N \parallel x_i$, run $\text{Rand}(M) = h^*$. If $h^* \oplus h = y_i$, return h and $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_i \rightarrow_{x_i} h^*\}$. If there is more than one such M we say the event **BAD** occurs and return randomly.
 - (b) Else return $P_{x_i}^{-1}(y_i) = h$ (say) and $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_i \rightarrow_{x_i} h^* \oplus h_i\}$.

The time complexity of the simulator is $O(lq^2)$. The worst case occurs to search all choices of $\mathbb{IV} \rightarrow_M h$ in the case of S^{-1} query. We define the **COLL** as defined in previously or **BAD** occurs. Let D be a distinguisher keeping relations \mathcal{R}_i and \mathcal{R}_i^* . Note that, $(E_x(y) = z \Leftrightarrow h \rightarrow_m h') \Leftrightarrow m = x, h = y$ and $h' = z \oplus y$. Now for a random permutation either z or y is chosen randomly.

1. For E query, define $S_i = \text{Supp}(\mathcal{R}_i) \oplus h_i \cup P_{x_i}[i]$, where $P_x[i]$ is the set of all images of P_x obtained from P_x or P_x^{-1} -query till i^{th} query of the distinguisher.
2. For E^{-1} query, $S_i = \text{Supp}(\mathcal{R}_i) \cup (\text{Supp}(\mathcal{R}_i) \oplus y_i) \cup P_{x_i}^{-1}[i]$, where $P_x^{-1}[i]$ is the set of all images of P_x^{-1} .
3. Define, $W_i = \{h \oplus h^* : \mathbb{IV} \rightarrow_M h \rightarrow_m h^* \in \mathcal{R}_{i-1}^* \text{ and } M \parallel m = g(X) \text{ for some } X\}$. This set is related to the **BAD** event.
4. Finally we define, $Z_i = S_i \cup W_i \cup \{h_i\}$ (for R query $h_i = \mathbb{IV}$, for E^{-1} query we can ignore $\{h_i\}$).

Now we say that \mathcal{V}_i is not collision-free if for for some $j \leq i$, the output of \mathcal{O}_2 oracle (in j^{th} query) is in W_i and it is not a trivial query. This definition is a modified definition of previous collision-free. Here we change the collision set to W_i . Similar to the previous results we have the following lemma and main theorem of this section.

Lemma 3. *The conditional distribution of H, E, E^{-1}, S and S^{-1} on i^{th} query, given $\mathcal{Q}_{i-1} \wedge \text{CF}$ is uniformly distributed on the set $X \setminus W_i$ provided it is not a trivial query, where $h_i = \text{IV}$ for \mathcal{O}_1 query or (h_i, x_i) be the query for \mathcal{O}_2 . In case of trivial query all distribution are degenerated.*

Proof. If the query is non-trivial query and collision free is true then $\text{Rand}, S, S^{-1}, E$ and E^{-1} are uniformly distributed on the set $X \setminus W_i$. In case of H^E , the hash function, we can prove that $\Pr[H^E(M_i) = a_1] = \Pr[H^E(M_i) = a_2]$, where $a_1, a_2 \in X \setminus W_i$. While we count all possible functions E , we interchange the roll of a_1 and a_2 in the inputs and outputs of E (as in H^f). We skip the detail of the proof as it is similar to Lemma 1.

If collision free is true the response of trivial query is completely determined by the past view (for all possible oracles). For example, if it is S^{-1} query then note that there are not more than one choice of M (or h , see case (1)) as BAD events is included in the event $\neg \text{CF}$. Thus, there is exactly one h which is completely determined by the past view and this is the response of this query. Other cases also can be checked. ■

Lemma 4. $\Pr[\neg \text{CF}_1] = O(\frac{l^2 q^2}{2^n})$ and $\Pr[\neg \text{CF}_2] = O(\frac{q^2}{2^n})$, where l is the maximum number of blocks in H -query.

Proof. The proof of the lemma is similar to lemma 2 except when BAD event occurs. For each query it will happen with probability $O(q/2^n)$ as $R(M) \oplus h = y_i$ has probability $1/2^n$ and there can be at most 2^n such M 's. ■

Theorem 5. *Prefix-free single length MD hash functions in a fixed-size random oracle model is (t_D, t_S, q, ϵ) -indifferentiable from a random oracle, for any t_D , with $t_S = l \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-n+1} \cdot l^2 \cdot \mathcal{O}(q^2)$, where l is the maximum length of a query made by the distinguisher D .*

Indifferentiability of Sixteen PGV Hash Functions

Now we consider all collision secure PGV hash functions. We will show, in the prefix-free mode, that sixteen (indexed by $1 \sim 16$ in table 1 of Appendix A) out of twenty are also indifferentiable with random oracle. Others (indexed by $17 \sim 20$ in table 1 of Appendix A) are not indifferentiable from random oracle. Till now we have shown for the case-1 of Appendix A. For other cases one can make similar analysis. For example, $h_i = f(h_{i-1}, m_i) = E_{w_i}(m_i) \oplus h_{i-1}$. So, $(E_k(x) = y \Leftrightarrow h \rightarrow_m h') \iff m = x, h = x \oplus k$ and $h' = k \oplus x \oplus y$. One can also define the simulator for other PGV functions similarly. The proof of the indifferentiability will follow similarly.

1. On query $(1, k_i, x_i)$ i.e. $E_{k_i}(x_i)$,
 - (a) If $\text{IV} \rightarrow_N h_i$ and $g(M) = N \parallel x_i$ then run $\text{Rand}(M)$ and obtain the response h^* . Return $h^* \oplus k_i \oplus x_i$ and $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(k_i \oplus x_i) \rightarrow_{x_i} h^*\}$ (otherwise behave randomly and similar to previous simulator this does not occur if collision-free occurs).
 - (b) Else return $P_{k_i}(x_i) = h^*, \mathcal{R}_i = \mathcal{R}_{i-1} \cup \{k_i \oplus x_i \rightarrow_{x_i} h^* \oplus k_i \oplus x_i\}$.

2. On query $(-1, k_i, y_i)$, i.e., $E_{k_i}^{-1}(y_i)$
 - (a) For each $\text{IV} \rightarrow_N h$ such that $g(M) = N \parallel k_i \oplus h$, run $\text{Rand}(M) = h^*$. If $h^* \oplus h = y_i$, return $h \oplus k_i$ and $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h \rightarrow_{k_i \oplus h} h^*\}$. If there is more than one such M we say the event **BAD** occurs and return randomly.
 - (b) Else return $F_{k_i}^{-1}(y_i) = h$ (say) and $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h \oplus k_i \rightarrow_{x_i} h^* \oplus h \oplus k_i\}$.

3.3 Indifferentiability of Double Length Hash Functions

Now we consider the double length construction. A $2n$ -bit hash value $x_l = (h_l, g_l)$ is computed from κl -bit message (m_1, m_2, \dots, m_l) as follows. For $i = 1, 2, \dots, l$, $F(x_{i-1}, m_i) = (h_i, g_i)$ such that

$$\begin{aligned} h_i &= f(h_{i-1}, g_{i-1}, m_i) \\ g_i &= f(p(h_{i-1}, g_{i-1}), m_i) \end{aligned}$$

where p is a permutation on $2n$ bits and p has no fixed point and $p(g, h) \neq (h, g)$ for any h, g . Further we assume that $p^2(\cdot)$ is an identity permutation. One example would be $p(x) = \bar{x}$, where \bar{x} is the bitwise complement. We define an equivalence relation, $w \equiv w^*$ if $w = p(w^*)$ or $w = w^*$. Like previous simulator we define the simulator as follows:

Simulator: The simulator keeps the relations $(\mathcal{R}_1, \dots, \mathcal{R}_{i-1})$. Initially, $\mathcal{R}_0 = \emptyset$. On the i^{th} query (h_i, g_i, x_i) , the response of S is as follow:

1. If the i^{th} query is same as a previous query, output same output of the previous query.
2. Else if $\exists \text{IV} \rightarrow_N h \parallel g \in \mathcal{R}_{i-1}, g(M) = N \parallel x_i$ where $h \parallel g \equiv h_i \parallel g_i$, then run $\text{Rand}(M)$ and obtain the response $h^* \parallel g^*$. For more than one choices of M , return a random string $h^* \parallel g^*$ (this will never happen if $(\mathcal{R}_1, \dots, \mathcal{R}_q)$ is collision-free).
 - (a) If $h \parallel g = h_i \parallel g_i$ then return h^* .
 - (b) If $h \parallel g = p(h_i \parallel g_i)$ then return g^* .
 - (c) If $(p(h_i \parallel g_i), x_i)$ has been queried before then
 - i. If $h \parallel g = h_i \parallel g_i$ then $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h \parallel g \rightarrow_{x_i} h^* \parallel g^*\}$.
 - ii. If $h \parallel g = p(h_i \parallel g_i)$ then $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h \parallel g \rightarrow_{x_i} g^* \parallel h^*\}$.
3. Else return a random string h^* . If $(p(h_i \parallel g_i), x_i)$ has been queried before and response is g^* then $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_i \parallel g_i \rightarrow_{x_i} h^* \parallel g^*\} \cup \{p(h \parallel g) \rightarrow_{x_i} g^* \parallel h^*\}$.

If distinguisher is making q queries at most then one can design the above simulator so that it runs in time $O(lq)$. In the worst case simulator has to back track to initial value to check whether condition (1) is satisfied or not and this needs at most $O(lq)$ time. Similar to previous results we have the following lemma and main theorem of this section. Similar to prefix free MD construction, we can define *support* and *collision free*.

1. Define *support* of a relation \mathcal{R}_i by a subset of X , $\text{Supp}(\mathcal{R}_i) = \{h \parallel g, p(h \parallel g) : h \parallel g \rightarrow_m h_1 \parallel g_1 \in \mathcal{R}_i\} \cup \{h \parallel g, p(h \parallel g) : h_1 \parallel g_1 \rightarrow_m h \parallel g \in \mathcal{R}_i\} \cup \{\text{IV}\}$. Note that, $\text{Supp}(\mathcal{R}_i) = \text{Supp}(\mathcal{R}_i^*)$.

2. We say, r_i is a *trivial query* if $r_i \in \mathcal{R}_{i-1}^*$. Since g is a prefix-free padding, r_i can be trivial query only if any one of the following holds :
 - (a) $r_i = \mathbf{IV} \rightarrow_{g(M)} h_\ell || g_\ell$, where $\mathbf{IV} = h_0 || g_0 \rightarrow_{m_1} h_1 || g_1 \rightarrow_{m_2} \dots h_{\ell-1} || g_{\ell-1} \rightarrow_{m_\ell} h_\ell || g_\ell \in \mathcal{R}_{i-1}^*$ and $g(M) = m_1 || \dots || m_\ell$.
 - (b) $r_i = h_{\ell-1} || g_{\ell-1} \rightarrow_{m_\ell} h_\ell$ or $p(h_{\ell-1} || g_{\ell-1}) \rightarrow_{m_\ell} g_\ell$, where $\mathbf{IV} = h_0 || g_0 \rightarrow_{m_1} h_1 || g_1 \rightarrow_{m_2} \dots h_{\ell-1} || g_{\ell-1}$, $\mathbf{IV} \rightarrow_{g(M)} h_\ell || g_\ell \in \mathcal{R}_{i-1}^*$ and $g(M) = m_1 || \dots || m_\ell$.
 - (c) r_i is a repetition query i.e. $r_i = r_j$ for some $j < i$. For simplicity we can assume that there is no repetition query as distinguisher's point of view it does not help anything.
3. We say \mathcal{V} is not *collision free* (or in short $\neg \text{CF}$) if for some i one of followings hold :
 - (a) In case of \mathcal{O}_1 query : $r_i = h_i || g_i \rightarrow_M h' || g'$ is non trivial and $h' || g' \in \text{Supp}(\mathcal{R}_{i-1}) \cup \{h_i || g_i\}$.
 - (b) In case of \mathcal{O}_2 query : $r_i = h_i || g_i \rightarrow_{m_i} h'$ is non trivial and $\mathcal{O}_2(p(h_i || g_i), x_i) = g'$ has been queried before and $h' || g'$ or $g' || h' \in \text{Supp}(\mathcal{R}_{i-1}) \cup \{h_i || g_i\} \cup \{p(h_i || g_i)\}$.

Lemma 5. *Given $\mathcal{Q}_{i-1} \wedge \text{CF}$, the conditional distribution of H, f, Rand and S on i^{th} query is uniformly distributed on the set $X \setminus (\text{Supp}(\mathcal{R}_{i-1}) \cup \{h_i || g_i\})$ provided it is not a trivial query, where $h_i || g_i = \mathbf{IV}$ for \mathcal{O}_1 query or $(h_i || g_i, x_i)$ be the query for \mathcal{O}_2 .*

Proof. Given $a (= a_1 || a_2)$ and $a' (= a'_1 || a'_2) \notin X \setminus (\text{Supp}(\mathcal{R}_{i-1}) \cup \{h_i || g_i\})$, Let $A = \{f : X \times B \rightarrow X : H^f(M) = a \wedge f \text{ satisfies } \mathcal{Q}_{i-1}\}$. Similarly define A' for a' . Similar to prefix free MD construction, we can define a bijection ϕ between A and A' similar to the Lemma 5.

1. If $f \in A$ then $\phi(f)(b, x) || \phi(f)(p(b), x) = f(b, x) || f(p(b), x)$ if $\{f(b, x) || f(p(b), x), b\} \cap \{a, a'\} = \emptyset$
2. $\phi(f)(a, x) || \phi(f)(p(a), x) = f(a', x) || f(p(a'), x)$ if $f(a', x) || f(p(a'), x) \notin \{a, a'\}$. Similarly, $\phi(f)(a', x) || \phi(f)(p(a'), x) = f(a, x) || f(p(a), x)$ if $f(a, x) || f(p(a), x) \notin \{a, a'\}$.
3. If $b \notin \{a, a'\}$ but $f(b, x) || f(p(b), x) = a$ then $\phi(f)(b, x) || \phi(f)(p(b), x) = a'$. Similarly, $f(b, x) || f(p(b), x) = a'$ then $\phi(f)(b, x) || \phi(f)(p(b), x) = a$.
4. There are four other possibilities i.e.
 - (a) if $f(a, x) || f(p(a), x) = a$ then $\phi(f)(a', x) || \phi(f)(p(a'), x) = a'$.
 - (b) if $f(a, x) || f(p(a), x) = a'$ then $\phi(f)(a', x) || \phi(f)(p(a'), x) = a$.
 - (c) if $f(a', x) || f(p(a'), x) = a$ then $\phi(f)(a, x) || \phi(f)(p(a), x) = a'$.
 - (d) if $f(a', x) || f(p(a'), x) = a'$ then $\phi(f)(a, x) || \phi(f)(p(a), x) = a$.

Now it is easy to check that $\phi(f)$ is well defined and it belongs to A' . Here, we mainly interchange the roll of a and a' in all possible cases of input and output keeping others same. Thus, given $H^f(M) = a$, we should have $H^{\phi(f)}(M) = a'$ keeping all other equalities fixed (in \mathcal{Q}_{i-1}). Now it is also easy to check that this is a bijection as we can define the inverse function similarly. Thus, $|A| = |A'|$

and hence the probabilities are equal. We can prove similarly for the distribution of f . So we skip the proof of this. \blacksquare

Now we bound the probability of collision events for both cases.

Lemma 6. $\Pr[\neg CF_1] = O(\frac{l^2 q^2}{2^{2n}})$ and $\Pr[\neg CF_2] = O(\frac{q^2}{2^{2n}})$, where l is the maximum number of blocks in H -query and $|X| = 2^{2n}$.

Proof. The proof is also similar to the Lemma 2. So we skip the proof. \blacksquare

Theorem 6. Let F be above double length hash function. Then for any prefix-free function g , MD_g^F in a single-size random oracle model is (t_D, t_S, q, ϵ) -indifferentiable from a random oracle, for any t_D , with $t_S = l \cdot \mathcal{O}(q)$ and $\epsilon = 2^{-2n+1} \cdot l^2 \cdot \mathcal{O}(q^2)$, where l is the maximum length of a query made by the distinguisher D .

4 Attack on Some SBL and DBL Hash Functions

In this section we define PGV and PBGV hash functions. We give some indifferentiable attacks on some of these hash functions. We show only attacks with one-block padded message. More than one block, we can attack similarly.

The Preneel-Govaerts-Vandewalle (PGV) Schemes [17]

Let x_0 be the initial value and $\kappa = N$. E is N -bit block cipher with an N -bit key. An N -bit hash value x_l is computed from κl -bit message (m_1, m_2, \dots, m_l) as follows. For $i = 1, 2, \dots, l$,

$$F(x_{i-1}, m_i) = x_i = E_a(b) \oplus c$$

where $a, b, c \in \{x_{i-1}, m_i, v, x_{i-1} \oplus m_i\}$. Here, v is a constant.

Among 20 collision resistant PGV schemes, even we use prefix-free padding g , we show that 4 schemes are differentiable from random oracle. 4 schemes are $F_1(h_{i-1}, m_i) = E_{h_{i-1}}(m_i) \oplus m_i$, $F_2(h_{i-1}, m_i) = E_{h_{i-1}}(m_i \oplus h_{i-1}) \oplus m_i \oplus h_{i-1}$, $F_3(h_{i-1}, m_i) = E_{h_{i-1}}(m_i) \oplus m_i \oplus h_{i-1}$, and $F_4(h_{i-1}, m_i) = E_{h_{i-1}}(m_i \oplus h_{i-1}) \oplus m_i$. Here, we consider F_1 . Similarly, we can show the insecurity of other 3 cases.

- distinguisher D can access to oracles $(\mathcal{O}_1, \mathcal{O}_2)$ where $(\mathcal{O}_1, \mathcal{O}_2)$ is (H, E, E^{-1}) or (Rand, S, S^{-1}) .
 - make a random query M such that $g(M) = m$ and $|m| = n$. then give the query M to oracle \mathcal{O}_1 and receive z .
 - make an inverse query $(-1, x_0, z \oplus m)$ to \mathcal{O}_2 and receive m^* .
 - if $m = m^*$ output 1, otherwise 0.
 - Since any simulator S can know random m only with probability 2^{-n} ,

$$|\Pr[D^{H,E,E^{-1}} = 1] - \Pr[D^{R,S,S^{-1}} = 1]| = 1 - 2^{-n}$$

This is not negligible. So $MD_g^{F_1}$ is differentiable from random oracle.

The Preneel-Bosselaers-Govaerts-Vandewalle (PBGV) Scheme [16]

Let $x_0 = (h_0, g_0)$ be initial value and $N = 2n$ and $\kappa = N$. E is N -bit block cipher with an N -bit key. A N -bit hash value $x_l = (h_l, g_l)$ is computed from κl -bit message $m = (m_1, m_2, \dots, m_l)$ where $m_i = (m_{i,1}, m_{i,2})$ and $|m_{i,1}| = |m_{i,2}| = n$. For $i = 1, 2, \dots, l$, $F(x_{i-1}, m_i) = (h_i, g_i)$ is defined as follows.

$$\begin{aligned}
 h_i &= E_{m_{i,1} \oplus m_{i,2}}(h_{i-1} \oplus g_{i-1}) \oplus m_{i,1} \oplus h_{i-1} \oplus g_{i-1} \\
 g_i &= E_{m_{i,1} \oplus h_{i-1}}(m_{i,2} \oplus g_{i-1}) \oplus m_{i,2} \oplus h_{i-1} \oplus g_{i-1}
 \end{aligned}$$

The following is the indiffereniable attack for the PBGV scheme.

- distinguisher D can access to oracles $(\mathcal{O}_1, \mathcal{O}_2)$ where $(\mathcal{O}_1, \mathcal{O}_2)$ is (H, E, E^{-1}) or (Rand, S, S^{-1}) .
 - make a random query M such that $g(M) = m_1 = m_{1,1} || m_{1,2}$ and $|m_1| = 2n$. Then give the query M to oracle \mathcal{O}_1 and receive $x_1 = (h_1, g_1)$.
 - make an inverse query $(-1, m_{1,2} \oplus h_0 \oplus g_0 \oplus g_1, m_{1,1} \oplus h_0)$ to \mathcal{O}_2 and receive *out*.
 - if *out* = $m_{1,2} \oplus g_0$ output 1, otherwise 0.
 - Since any simulator S can know random $m_{1,2}$ only with probability 2^{-n} ,

$$|\Pr[D^{H,E,E^{-1}} = 1] - \Pr[D^{R,S,S^{-1}} = 1]| = 1 - 2^{-n}$$

This is not negligible. So MD_g^F is differentiable from random oracle.

By using the same idea one can find indiffereniable attack on QG-I, LOKI DBH, MDC-2 and some of the Hirose’s double length hash constructions.

5 Conclusion

As hash function is at times a popular candidate for approximation of a random oracle, the notion of indiffereniable is important to study. In this paper we have studied many known designs of hash function in term of indiffereniable. Some of them are secure and against some of them we have found attack. So there are many designs, for example sixteen PGV hash functions, which are secure beyond the collision security. This paper also presents an unified way to prove the indiffereniable for many designs of hash functions. Finally we note that there are still many designs whose security analysis in the view of indiffereniable are open.

Acknowledgement

We wish to thank Professor Douglas R. Stinson who helped us to get an idea of proving the result. The first author was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (KRF- 2005-213-C00005).

References

1. M. Bellare and P. Rogaway. Random Oracles Are Practical : A Paradigm for Designing Efficient Protocols. In *1st Conference on Computing and Communications Security*, ACM, pages 62–73. 1993.
2. J. Black, P. Rogaway, and T. Shrimpton. Black-box analysis of the block-cipher-based hash function constructions from PGV. In *Advances in Cryptology-Crypto'2002*, volume **2442** of *Lecture Notes in Computer Science*, pages 320–335. Springer-Verlag, 2002.
3. B. O. Brachtel, D. Coppersmith, M. M. Hyden, S. M. Matyas, C. H. Meyer, J. Oseas, S. Pilpel, and M. Schilling, "Data authentication using modification detection codes based on a public one way encryption function ," U.S. Patent Number 4,908,861, March 13, 1990.
4. L. Brown, J. Pieprzyk and J. Seberry. LOKI - A Cryptographic Primitive for Authentication and Secrecy Applications. In *Advances in Cryptology-Auscrypt'1990*, volume **453** of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 1990.
5. J. S. Coron, Y. Dodis, C. Malinaud and P. Puniya. Merkle-Damgard Revisited: How to Construct a Hash Function. In *Advances in Cryptology-Crypto'2005*, volume **3621** of *Lecture Notes in Computer Science*, pages 430–448. Springer-Verlag, 2005.
6. I. B. Damgard. A design principle for hash functions. In *Advances in Cryptology-Crypto'1989*, volume **435** of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1989.
7. S. Even, and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. In *Advances in Cryptology-Asiacrypt'1991*, volume **739** of *Lecture Notes in Computer Science*, pages 210–224. Springer-Verlag, 1992.
8. S. Hirose. Provably Secure Double-Block-Length Hash Functions in a Black-Box Model. In *ICISC'2004*, volume **3506** of *Lecture Notes in Computer Science*, pages 330–342. Springer-Verlag, 2005.
9. S. Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. To appear in proceeding of FSE' 2006.
10. J. Kilian, and P. Rogaway. How to protect DES against exhaustive key search. In *Journal of Cryptology*, **14(1)**:17-35, 2001, Earlier version in CRYPTO' 96.
11. X. Lai and J. L. Massey. Hash Functions Based on Block Ciphers. In *Advances in Cryptology-Eurocrypt'1992*, volume **658** of *Lecture Notes in Computer Science*, pages 55–70. Springer-Verlag, 1993.
12. Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In *Advances in Cryptology-Asiacrypt'2005*, volume **3788** of *Lecture Notes in Computer Science*, pages 474–494. Springer-Verlag, 2005.
13. U. Maurer, R. Renner and C. Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In *TCC'2004*, volume **2951** of *Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, 2004.
14. R. C. Merkle. One way hash functions and DES. In *Advances in Cryptology-Crypto'1989*, volume **435** of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1990.
15. Mridul Nandi. Towards Optimal Double-Length Hash Functions. In *Indocrypt'2005*, volume **3797** of *Lecture Notes in Computer Science*, pages 77–89. Springer-Verlag, 2005.

16. B. Preneel, A. Bosselaers, R. Govaerts and J. Vandewalle. Collision-free Hash-functions Based on Blockcipher Algorithms. Proceedings of 1989 International Carnahan Conference on Security Technology, pages 203–210.
17. B. Preneel, R. Govaerts and J. Vandewalle. Hash Functions based on Block Ciphers : A Synthetic approach. In *Advances in Cryptology-Crypto'1993*, volume **773** of *Lecture Notes in Computer Science*, pages 368–378. Springer-Verlag, 1994.
18. J. J. Quisquater and M. Girault. 2n-bit Hash Functions Using n-bit Symmetric Block Cipher Algorithms. In *Advances in Cryptology-Eurocrypt'1989*, volume **434** of *Lecture Notes in Computer Science*, pages 102–109. Springer-Verlag, 1990.
19. C. Shannon. Communication theory of secrecy systems. Bell Systems Technical Journal, **28**(4): pages 656–715, 1949.
20. R. Winternitz. A secure one-way hash function built from DES. In *Proceedings of the IEEE Symposium on Information Security and Privacy*, pages 88–90, 1984.

Appendix A: Table of Twenty PGV Hash Functions

Table 1. 20 Collision Resistant PGV Hash Functions in the Ideal Cipher Model. ($w_i = m_i \oplus h_{i-1}$).

Case	PGV	Case	PGV
1	$E_{m_i}(h_{i-1}) \oplus h_{i-1}$	11	$E_{m_i}(h_{i-1}) \oplus v$
2	$E_{m_i}(w_i) \oplus w_i$	12	$E_{w_i}(h_{i-1}) \oplus v$
3	$E_{m_i}(h_{i-1}) \oplus w_i$	13	$E_{m_i}(h_{i-1}) \oplus m_i$
4	$E_{m_i}(w_i) \oplus h_{i-1}$	14	$E_{w_i}(h_{i-1}) \oplus w_i$
5	$E_{w_i}(m_i) \oplus m_i$	15	$E_{m_i}(w_i) \oplus v$
6	$E_{w_i}(h_{i-1}) \oplus h_{i-1}$	16	$E_{m_i}(w_i) \oplus m_i$
7	$E_{w_i}(m_i) \oplus h_{i-1}$	17	$E_{h_{i-1}}(m_i) \oplus m_i$
8	$E_{w_i}(h_{i-1}) \oplus m_i$	18	$E_{h_{i-1}}(w_i) \oplus w_i$
9	$E_{w_i}(w_i) \oplus v$	19	$E_{h_{i-1}}(m_i) \oplus w_i$
10	$E_{w_i}(m_i) \oplus w_i$	20	$E_{h_{i-1}}(w_i) \oplus m_i$

Multi-Property-Preserving Hash Domain Extension and the EMD Transform

Mihir Bellare and Thomas Ristenpart

Dept. of Computer Science & Engineering 0404, University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404, USA
{mihir, tristenp}@cs.ucsd.edu
<http://www-cse.ucsd.edu/users/{mihir,tristenp}>

Abstract. We point out that the seemingly strong *pseudorandom oracle preserving* (PRO-Pr) property of hash function domain-extension transforms defined and implemented by Coron et. al. [1] can actually *weaken* our guarantees on the hash function, in particular producing a hash function that fails to be even collision-resistant (CR) even though the compression function to which the transform is applied is CR. Not only is this true in general, but we show that *all* the transforms presented in [1] have this weakness. We suggest that the appropriate goal of a domain extension transform for the next generation of hash functions is to be multi-property preserving, namely that one should have a *single* transform that is simultaneously at least collision-resistance preserving, pseudorandom function preserving and PRO-Pr. We present an efficient new transform that is proven to be multi-property preserving in this sense.

Keywords: Hash functions, random oracle, Merkle-Damgård, collision-resistance, pseudorandom function.

1 Introduction

BACKGROUND. Recall that hash functions are built in two steps. First, one designs a compression function $h: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$, where d is the length of a data block and n is the length of the chaining variable. Then one specifies a *domain extension transform* H that utilizes h as a black box to implement the hash function $H^h: \{0, 1\}^* \rightarrow \{0, 1\}^n$ associated to h . All widely-used hash functions use the Merkle-Damgård (MD) transform [2,3] because it has been proven [2,3] to be *collision-resistance preserving* (CR-Pr): if h is collision-resistant (CR) then so is H^h . This means that the cryptanalytic validation task can be confined to the compression function.

A RISING BAR. Current usage makes it obvious that CR no longer suffices as the security goal for hash functions. In order to obtain MACs and PRFs, hash functions were keyed. The canonical construct in this domain is HMAC [4,5] which is widely standardized and used. (NIST FIPS 198, ANSI X9.71, IETF RFC 2104, SSL, SSH, IPSEC, TLS, IEEE 802.11i, and IEEE 802.16e are only

some instances.) Hash functions are also used to instantiate random oracles [6] in public-key schemes such as RSA-OAEP [7] and RSA-PSS [8] in the RSA PKCS#1 v2.1 standard [9]. CR is insufficient for arguing the security of hash function based MACs or PRFs, let alone hash-function based random oracles. And it does not end there. Whether hash function designers like it or not, application builders will use hash functions for all kinds of tasks that presume beyond-CR properties. Not all such uses can be sanctified, but the central and common ones should be. We think that the type of usage we are seeing for hash functions will continue, and it is in the best interests of security to make the new hash functions rise as far towards this bar as possible, by making them strong and versatile tools that have security attributes beyond CR.

THIS PAPER. Towards the goal of building strong, multi-purpose hash functions, our focus is on domain extension, meaning we wish to determine which domain extension transforms are best suited to this task. The first part of our work examines a natural candidate, namely transforms that are *pseudorandom oracle preserving* as per [1], and identifies some weaknesses of this goal. This motivates the second part, where we introduce the notion of a *multi-property preserving (MPP) transform*, argue that this should be the target goal, and present and prove the correctness of an efficient MPP transform that we refer to as EMD. Let us now look at all this in more depth.

RANDOM-ORACLE PRESERVATION. Coron, Dodis, Malinaud and Puniya [1] make the important observation that random oracles are modeled as monolithic entities (i.e., are black boxes working on domain $\{0, 1\}^*$), but in practice are instantiated by hash functions that are highly structured due to the design paradigm described above, leading for example to the extension attack. Their remedy for this logical gap is to suggest that a transform H be judged secure if, when modeling h as a fixed-input-length random oracle, the resulting scheme H^h behaves like a random oracle. They give a formal definition of “behaving like a random oracle” using the indistinguishability framework of Maurer et al. [10]. We use the moniker *pseudorandom oracle* to describe any construction that is indistinguishable from a random oracle. (Note that a random oracle itself is always a pseudorandom oracle.) The framework has the desirable property that any scheme proven secure in the random oracle model of [6] is still secure when we replace the random oracles with pseudorandom oracles. We call the new security goal of [1] *pseudorandom oracle preservation (PRO-Pr)*. They propose four transforms which they prove to be PRO-Pr.

PRO-Pr seems like a very strong property to have. One reason one might think this is that it appears to automatically guarantee that the constructed hash function has many nice properties. For example, that the hash function created by a PRO-Pr transform would be CR. Also that the hash function could be keyed in almost any reasonable way to yield a PRF and MAC. And so on. This would be true, because random oracles have these properties, and hence so do pseudorandom oracles. Thus, one is lead to think that one can stop with PRO-Pr: once the transform has this property, we have all the attributes we desire from the constructed hash function.

WEAKNESS OF PRO-Pr. The first contribution of this paper is to point out that the above reasoning is flawed and there is a danger to PRO-Pr in practice. Namely, the fact that a transform is PRO-Pr *does not guarantee* that the constructed hash function is CR, even if the compression function is CR. We demonstrate this with a counter-example. Namely we give an example of a transform that is PRO-Pr, yet there is a CR compression function such that the hash function resulting from the transform is not CR. That is, the transform is PRO-Pr but not CR-Pr, or, in other words, PRO-Pr does not imply CR-Pr. What this shows is that using a PRO-Pr transform could be *worse* than using the standard, strengthened Merkle-Damgård transform from the point of view of security because at least the latter guarantees the hash function is CR if the compression function is, but the former does not. If we blindly move to PRO-Pr transforms, our security guarantees are actually going down, not up.

How can this be? It comes about because PRO-Pr provides guarantees only if the compression function is a random oracle or pseudorandom oracle. But of course any real compression function is *provably not* either of these. (One can easily differentiate it from a random oracle because it can be computed by a small program.) Thus, when a PRO-Pr transform works on a real compression function, we have essentially no provable guarantees on the resulting hash function. This is in some ways analogous to the kinds of issues pointed out in [11,12] about the sometimes impossibility of instantiating random oracles.

THE TRANSFORMS OF [1] ARE NOT CR-Pr. The fact that a PRO-Pr transform need not in general be CR-Pr does not mean that some *particular* PRO-Pr transform is not CR-Pr. We therefore investigate each of the four PRO-Pr schemes suggested by [1]. The schemes make slight modifications to the MD transform: the first applies a prefix-free encoding, the second “throws” away some of the output, and the third and fourth utilize an extra compression function application. Unfortunately, we show that none of the four transforms is CR-Pr. We do this by presenting an example CR compression function h such that applying each of the four transforms to it results in a hash function for which finding collisions is trivial. In particular, this means that these transforms do not provide the same guarantee as the existing and in-use Merkle-Damgård transform. For this reason we think these transforms should not be considered suitable for use in the design of new hash functions.

WHAT THIS MEANS. We clarify that we are *not* suggesting that the pseudorandom oracle preservation goal of [1] is unimportant or should not be achieved. In fact we think it is a very good idea and should be a property of any new transform. This is so because in cases where we are (heuristically) assuming the hash function is a random oracle, this goal reduces the assumption to the compression function being a random oracle. What we have shown above, however, is that *by itself*, it is not enough because it can weaken existing, standard-model guarantees. This leads to the question of what exactly *is* enough, or what we should ask for in terms of a goal for hash domain extension transforms.

MPP TRANSFORMS. The two-step design paradigm in current use is compelling because it reduces the cryptanalytic task of providing CR of the hash function to certifying only that the compression function has the same property. It makes sense to seek other attributes via the appropriate extension of this paradigm. We suggest that, if we want a hash function with properties P_1, \dots, P_n then we should (1) design a compression function h with the goal of having properties P_1, \dots, P_n , and (2) apply a domain extension transform H that *provably* preserves P_i for every $i \in [1..n]$. We call such a compression function a multi-property one, and we call such a transform a *multi-property-preserving domain extension transform* (from now on simply an MPP transform). Note that we want a *single* transform that preserves multiple properties, resulting in a single, multi-property hash function, as opposed to a transform per property which would result in not one but numerous hash functions. We suggest that multi-property preservation is the goal a transform should target.

PROPERTIES TO PRESERVE. Of course the next question to ask is which properties our MPP domain extension transform should preserve. We wish, of course, that the transform continue to be CR-Pr, meaning that it preserve CR. The second thing we ask is that it be pseudorandom function preserving (PRF-Pr). That is, if an appropriately keyed version of the compression function is a PRF then the appropriately keyed version of the hash function must be a PRF too. This goal is important due to the many uses of hash functions as MACs and PRFs via keying as mentioned above. Indeed, if we have a compression function that can be keyed to be a PRF and our transform is PRF-Pr then obtaining a PRF or MAC from a hash function will be simple and the construction easy to justify. The final goal we will ask is that the transform be PRO-Pr. Compelling arguments in favor of this goal were made at length in [1] and briefly recalled above.

To be clear, we ask that, for a transform H to be considered suitable, one should do the following. First, prove that H^h is CR using only the fact that h is CR. Then show that H^h is a pseudorandom oracle when h is a pseudorandom oracle. Finally, use some natural keying strategy to key H^h and assume that h is a good PRF, then prove that H^h is also a good PRF. We note that such a MPP transform will not suffer from the weakness of the transforms of [1] noted above because it will be not only PRO-Pr but also CR-Pr and PRF-Pr.

NEW TRANSFORM. There is to date no transform with all the properties above. (Namely, that it is PRO-Pr, CR-Pr and PRF-Pr.) The next contribution of this paper is a new transform EMD (Enveloped Merkle-Damgård) which is the first to meet our definition of hash domain extension security: EMD is proven to be CR-Pr, PRO-Pr, and PRF-Pr. The transform is simple and easy to implement in practice (see the figure in Section 5). It combines two mechanisms to ensure that it preserves all the properties of interest. The first mechanism is the well-known Merkle-Damgård strengthening [2]: we always concatenate an input message with the 64-bit encoding of its length. This ensures that EMD is CR-Pr. The second mechanism is the use of an “envelope” to hide the internal MD iteration — we apply the compression function in a distinguished way to the output of the plain MD iteration. Envelopes in this setting were previously used by the NMAC and

Transform	CR-Pr	PRO-Pr	PRF-Pr	Uses of h for $ M = b \geq d$
Plain MD (MD)	No	No	No	$\lceil (b+1)/d \rceil$
Strengthened MD (SMD)	[2,3]	No	No	$\lceil (b+1+64)/d \rceil$
Prefix-Free (PRE)	No	[1]	[13]	$\lceil (b+1)/(d-1) \rceil$
Chop Solution (CHP)	No	[1]	?	$\lceil (b+1)/d \rceil$
NMAC Construction (NT)	No	[1]	?	$1 + \lceil (b+1)/d \rceil$
HMAC Construction (HT)	No	[1]	?	$2 + \lceil (b+1)/d \rceil$
Enveloped MD (EMD)	[2]	Thm. 1	Thm. 2	$\lceil (b+1+64+n)/d \rceil$

Fig. 1. Comparison of transform security and efficiency when applied to a compression function $h: \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$. The last column specifies the number of calls to h needed to hash a b -bit message M (where $b \geq d$) under each transform and a typical padding function (which minimally adds a bit of overhead).

HMAC constructions [4] to build PRFs out of compression functions, and again in two of the PRO-Pr transforms of [1], which were also based on NMAC and HMAC. We utilize the envelope in a way distinct from these prior constructions. Particularly, we include message bits as input to the envelope, which increases the efficiency of the scheme. Second, we utilize a novel reduction technique in our proof that EMD is PRO-Pr to show that simply fixing n bits of the envelope’s input is sufficient to cause the last application of the random oracle to behave independently with high probability. This simple solution allows our transform to be PRO-Pr using a single random oracle without using the other work-arounds previously suggested (e.g., prefix-free encodings or prepending a block of zeros to input messages). A comparison of various transforms is given in Fig. 1.

PATCHING EXISTING TRANSFORMS. We remark that it is possible to patch the transforms of [1] so that they are CR-Pr. Namely, one could use Merkle-Damgård strengthening, which they omitted. However our transform still has several advantages over their transforms. One is that ours is cheaper, i.e. more efficient, and this matters in practice. Another is that ours is PRF-Pr. A result of [13] implies that one of the transforms of [1] is PRF-Pr, but whether or not this is true for the others is not clear.

WHENCE THE COMPRESSION FUNCTION? We do not address the problem of constructing a multi-property compression function. We presume that this can and will be done. This assumption might seem questionable in light of the recent collision-finding attacks [14,15] that have destroyed some hash functions and tainted others. But we recall that for block ciphers, the AES yielded by the NIST competition was not only faster than DES but seems stronger and more elegant. We believe it will be the same for compression functions, namely that the planned NIST hash function competition will lead to compression functions having the properties (CR and beyond) that we want, and perhaps without increase, or even with decrease, in cost, compared to current compression functions. We also note that we are not really making new requirements on the compression function; we are only making explicit requirements that are implicit even in current usage.

FAMILIES OF COMPRESSION FUNCTIONS. Several works [16,17,18] consider a setting where compression and hash functions are families rather than individual functions, meaning, like block ciphers, have an extra, dedicated key input. In contrast, we, following [4,1,5], adopt the setting of current practical cryptographic compression and hash functions where there is no such dedicated key input. An enveloping technique similar to that of EMD is used in the Chain-Shift construction of Maurer and Sjödin [18] for building a VIL MAC out of a FIL MAC in the dedicated key input setting. We further discuss this setting, and their work, in the full version of the paper [19].

2 Definitions

NOTATION. Let $D = \{0, 1\}^d$ and $D^+ = \cup_{i \geq 1} \{0, 1\}^{id}$. We denote pairwise concatenation by $\|$, e.g. $M \| M'$. We will often write the concatenation of a sequence of string by $M_1 \cdots M_k$, which translates to $M_1 \| M_2 \| \dots \| M_k$. For brevity, we define the following semantics for the notation $M_1 \cdots M_k \stackrel{\leftarrow}{\leftarrow} M$ where M is a string of $|M|$ bits: 1) define $k = \lceil |M|/d \rceil$ and 2) if $|M| \bmod d = 0$ then parse M into M_1, M_2, \dots, M_k where $|M_i| = d$ for $1 \leq i \leq k$, otherwise parse M into $M_1, M_2, \dots, M_{k-1}, M_k$ where $|M_i| = d$ for $1 \leq i \leq k-1$ and $|M_k| = |M| \bmod d$. For any finite set S we write $s \stackrel{\$}{\leftarrow} S$ to signify uniformly choosing a value $s \in S$.

ORACLE TMS, RANDOM ORACLES, AND TRANSFORMS. Cryptographic schemes, adversaries, and simulators are modeled as Oracle Turing Machines (OTM) and are possibly given zero or more oracles, each being either a random oracle or another OTM (note that when used as an oracle, an OTM maintains state across queries). We allow OTMs to expose a finite number of interfaces: an OTM $N = (N_1, N_2, \dots, N_l)$ exposes interfaces N_1, N_2, \dots, N_l . For brevity, we write M^N to signify that M gets to query all the interfaces of N . For a set Dom and finite set Rng we define a *random function* by the following TM accepting inputs $X \in Dom$:

```

Algorithm  $RF_{Dom, Rng}(X)$ :
  if  $T[X] = \perp$  then  $T[X] \stackrel{\$}{\leftarrow} Rng$ 
  ret  $T[X]$ 

```

where T is a table everywhere initialized to \perp . This implements a random function via lazy sampling (which allows us to reason about the case in which Dom is infinite). In the case that $Dom = \{0, 1\}^d$ and $Rng = \{0, 1\}^r$ we write $RF_{d,r}$ in place of $RF_{Dom, Rng}$. We similarly define $RF_{d, Rng}$ and $RF_{Dom, r}$ in the obvious ways and write $RF_{*,r}$ in the special case that $Dom = \{0, 1\}^*$. A *random oracle* is simply a public random function: all parties (including the adversary) are given access. We write $f, g, \dots = RF_{Dom, Rng}$ to signify that f, g, \dots are independent random oracles from Dom to Rng . A *transform* C describes how to utilize an arbitrary compression function to create a variable-input-length hash function. When we fix a particular compression function f , we get the associated cryptographic scheme $C^f \equiv C[f]$.

COLLISION RESISTANCE. We consider a function F to be collision resistant (CR) if it is computationally infeasible to find any two messages $M \neq M'$ such that $F(M) = F(M')$. For the rest of the paper we use h to always represent a collision-resistant compression function with signature $h: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$.

Note our definition of CR is informal. The general understanding in the literature is that a formal treatment requires considering keyed families. But practical compression and hash functions are not keyed when used for CR. (They can be keyed for use as MACs or PRFs.) And in fact, our results on CR are still formally meaningful because they specify explicit reductions.

PRFs. Let $F: Keys \times Dom \rightarrow Rng$ be a function family. Informally, we consider F a pseudorandom function family (PRF) if no reasonable adversary can succeed with high probability at distinguishing between $F(K, \cdot)$ for $K \xleftarrow{\$} Keys$ and a random function $f = RF_{Dom, Rng}$. More compactly we write the *prf-advantage* of an adversary A as

$$Adv_F^{prf}(A) = \Pr \left[K \xleftarrow{\$} Keys; A^{F(K, \cdot)} \Rightarrow 1 \right] - \Pr \left[A^{f(\cdot)} \Rightarrow 1 \right]$$

where the probability is taken over the random choice of K and the coins used by A or by the coins used by f and A . For the rest of the paper we use e to always represent a PRF with signature $e: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$ that is keyed through the low n bits of the input.

PROs. The indistinguishability framework [10] generalizes the more typical indistinguishability framework (e.g., our definition of a PRF above). The new framework captures the necessary definitions for comparing an object that utilizes public components (e.g., fixed-input-length (FIL) random oracles) with an ideal object (e.g., a variable-input-length (VIL) random oracle). Fix some number l . Let $C^{f_1, \dots, f_l}: Dom \rightarrow Rng$ be a function for random oracles $f_1, \dots, f_l = RF_{D, R}$. Then let $S^{\mathcal{F}} = (S_1, \dots, S_l)$ be a simulator OTM with access to a random oracle $\mathcal{F} = RF_{Dom, Rng}$ and which exposes interfaces for each random oracle utilized by C . (The simulator's goal is to mimic f_1, \dots, f_l in such a way as to convince an adversary that \mathcal{F} is C .) The *pro-advantage* of an adversary A against C is the difference between the probability that A outputs a one when given oracle access to C^{f_1, \dots, f_l} and f_1, \dots, f_l and the probability that A outputs a one when given oracle access to \mathcal{F} and $S^{\mathcal{F}}$. More succinctly we write that the pro-advantage of A is

$$Adv_{C, S}^{pro}(A) = \left| \Pr \left[A^{C^{f_1, \dots, f_l}, f_1, \dots, f_l} \Rightarrow 1 \right] - \Pr \left[A^{\mathcal{F}, S^{\mathcal{F}}} \Rightarrow 1 \right] \right|$$

where, in the first case, the probability is taken over the coins used by the random oracles and A and, in the second case, the probability is over the coins used by the random oracles, A , and S . For the rest of the paper we use f to represent a random oracle $RF_{d+n, n}$.

RESOURCES. We give concrete statements about the advantage of adversaries using certain resources. For prf-adversaries we measure the total number of queries q made and the running time t . For pro-adversaries we measure the total number of *left queries* q_L (which are either to C or \mathcal{F}) and the number of

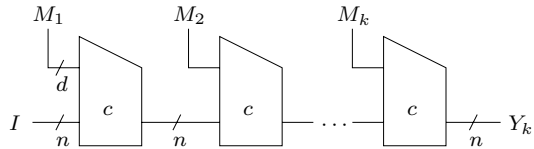
right queries q_i made to each oracle f_i or simulator interface S_i . We also specify the resources utilized by simulators. We measure the total number of queries q_S to \mathcal{F} and the maximum running time t_S . Note that these values are generally functions of the number of queries made by an adversary (necessarily so, in the case of t_S).

POINTLESS QUERIES. In all of our proofs (for all notions of security) we assume that adversaries make no *pointless queries*. In our setting this particularly means that adversaries are never allowed to repeat a query to an oracle.

3 Domain Extension Using Merkle-Damgård

THE MERKLE-DAMGÅRD TRANSFORM. We focus on variants of the Merkle-Damgård transform. Let $c: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$ be an arbitrary fixed-input-length function. Using it, we wish to construct a family of variable-input-length functions $F^c: \{0, 1\}^n \times \{0, 1\}^* \rightarrow \{0, 1\}^n$. We start by defining the Merkle-Damgård iteration $c^+: D^+ \rightarrow \{0, 1\}^n$ by the algorithm specified below.

Algorithm $c^+(I, M)$:
 $M_1 \cdots M_k \stackrel{d}{\leftarrow} M; Y_0 \leftarrow I$
for $i = 1$ **to** k **do**
 $Y_i \leftarrow c(M_i \parallel Y_{i-1})$
ret Y_k



Since I is usually fixed to a constant, the function c^+ only works for strings that are a multiple of d bits. Thus we require a padding function $\text{pad}(M)$, which for any string $M \in \{0, 1\}^*$ returns a string Y for which $|Y|$ is a multiple of d . We require that pad is one-to-one (this requirement is made for all padding functions in this paper). A standard instantiation for pad is to append to the message a one bit and then enough zero bits to fill out a block. Fixing some $IV \in \{0, 1\}^n$, we define the *plain Merkle-Damgård transform* $\text{MD}[c] = c^+(IV, \text{pad}(\cdot))$.

KEYING STRATEGIES. In this paper we discuss transforms that produce keyless schemes. We would also like to utilize these schemes as variable-input-length PRFs, but this requires that we use some keying strategy. We focus on the *key-via-IV strategy*. Under this strategy, we replace constant initialization vectors with randomly chosen keys of the same size. For example, if e is a particular PRF, then keyed MD^e would be defined as $\text{MD}_K^e(M) = e^+(K, \text{pad}(M))$ (it should be noted that this is not a secure PRF). We will always signify the keyed version of a construction by explicitly including the keys as subscripts.

MULTI-PROPERTY PRESERVATION. We would like to reason about the security of MD and its variants when we make assumptions about c . Phrased another way, we want to know if a transform such as MD *preserves* security properties of the underlying compression function. We are interested in collision-resistance preservation, PRO preservation, and PRF preservation. Let C be a transform that works on functions from $\{0, 1\}^{d+n}$ to $\{0, 1\}^n$. Let $h: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$

be a collision-resistant hash function. Then we say that C is *collision-resistance preserving* (CR-Pr) if the scheme C^h is collision-resistant. Let $f = \text{RF}_{d+n,n}$ be a random oracle. Then we say that C is *pseudorandom oracle preserving* (PRO-Pr) if the scheme C^f is a pseudorandom oracle. Let $e: \{0,1\}^{d+n} \rightarrow \{0,1\}^n$ be an arbitrary PRF (keyed via the low n bits). Then we say that C is *pseudorandom function preserving* (PRF-Pr) if the keyed-via-IV scheme C_K^e is a PRF. A transform for which all of the above holds is considered *multi-property preserving*.

SECURITY OF MD AND SMD. It is well known that MD is neither CR-Pr, PRO-Pr, or PRF-Pr [2,3,13,1]. The first variant that was proven CR-Pr was so-called MD with strengthening, which we denote by SMD. In this variant, the padding function is replaced by one with the following property: for M and M' with $|M| \neq |M'|$ then $M_k \neq M'_k$ (the last blocks after padding are distinct). A straightforward way to achieve a padding function with this property is to include an encoding of the message length in the padding. In many implementations, this encoding is done using 64 bits [20], which restricts the domain to strings of length no larger than 2^{64} . We therefore fix some padding function $\text{pad64}(M)$ that takes as input a string M and returns a string Y of length kd bits for some number k such that the last 64 bits of Y are an encoding of $|M|$. Using this padding function we define the *strengthened MD transform* $\text{SMD}[c] = c^+(IV, \text{pad64}(\cdot))$. We emphasize the fact that preservation of collision-resistance is *strongly dependent* on the choice of padding function. However, this modification to MD is alone insufficient for rendering SMD either PRF-Pr or PRO-Pr due to simple length-extension attacks [13,1].

4 Orthogonality of Property Preservation

In this section we illustrate that property preservation is orthogonal. Previous work [1] has already shown that collision-resistance preservation does not imply pseudorandom oracle preservation. We investigate the inverse: does a transform being PRO-Pr imply that it is also CR-Pr? We answer this in the negative by showing how to construct a PRO-Pr transform that is not CR-Pr. While this result is sufficient to refute the idea that PRO-Pr is a stronger security goal for transforms, it does not necessarily imply anything about specific PRO-Pr transforms. Thus, we investigate the four transforms proposed by Coron et al. and show that all four fail to preserve collision-resistance. Finally, lacking a formally meaningful way of comparing pseudorandom oracle preservation and pseudorandom function preservation (one resulting in keyless schemes, the other in keyed), we briefly discuss whether the proposed transforms are PRF-Pr.

4.1 PRO-Pr Does Not Imply CR-Pr

Let $n, d > 0$ and $h: \{0,1\}^{d+n} \rightarrow \{0,1\}^n$ be a collision-resistant hash function and $f = \text{RF}_{d+n,n}$ be a random oracle. Let Dom, Rng be non-empty sets and let C_1 be a transform for which $C_1^f \equiv C_1[f]$ is a pseudorandom oracle $C_1^f: Dom \rightarrow Rng$. We create a transform C_2 that is PRO-Pr but is *not* CR-Pr. In other words

procedure Initialize	procedure $C(X)$	Game G0	Game G1
000 $f = \text{RF}_{d+n,n}$	200 $Y \leftarrow C_1^f(X)$		
procedure $f(x)$	201 if $f(0^{d+n}) = 0^n$ then $\text{bad} \leftarrow \text{true};$	$Y \leftarrow 0^n$	
100 ret $f(x)$	202 ret Y		

Fig. 2. Games utilized in the proof of Proposition 1 to show that C_2^f is a PRO

the resulting scheme $C_2^f: \text{Dom} \rightarrow \text{Rng}$ is indifferentiable from a random oracle, but it is trivial to find collisions against the scheme C_2^h (even without finding collisions against h). We modify $C_1[c]$ to create $C_2[c]$ as follows. First check if $c(0^{d+n})$ is equal to 0^n and return 0^n if that is the case. Otherwise we just follow the steps specified by $C_1[c]$. Thus the scheme C_2^f returns 0^n for any message if $f(0^{d+n}) = 0^n$. Similarly the scheme C_2^h returns 0^n for any message if $h(0^{d+n}) = 0^n$. The key insight, of course, is that the differing assumptions made about the oracle impact the likelihood of this occurring. If the oracle is a random oracle, then the probability is small: we prove below that C_2^f is a pseudorandom oracle. On the other hand, we now show how to easily design a collision-resistant hash function h that causes C_2^h to not be collision resistant. Let $h': \{0, 1\}^{d+n} \rightarrow \{0, 1\}^{n-1}$ be some collision-resistant hash function. Then $h(M)$ returns 0^n if $M = 0^{d+n}$, otherwise it returns $h'(M) || 1$. Collisions found on h would necessarily translate into collisions for h' , which implies that h is collision-resistant. Furthermore since $h(0^{d+n}) = 0^n$ we have that $C_2^h(M) = 0^n$ for any message M , making it trivial to find collisions against C_2^h .

Proposition 1. [C_2 is PRO-Pr] *Let $n, d > 0$ and Dom, Rng be non-empty sets and $f = \text{RF}_{d+n,n}$ and $\mathcal{F} = \text{RF}_{\text{Dom}, \text{Rng}}$ be random oracles. Let C_1^f be a pseudorandom oracle. Let C_2^f be the scheme as described above and let S be an arbitrary simulator. Then for any adversary A_2 that utilizes q_L left queries, q_R right queries, and runs in time t , there exists an adversary A_1 such that*

$$\text{Adv}_{C_2, S}^{\text{PRO}}(A_2) \leq \text{Adv}_{C_1, S}^{\text{PRO}}(A_1) + \frac{1}{2^n}.$$

with A_1 utilizing the same number of queries and time as A_2 .

Proof. Let $f = \text{RF}_{d+n,n}$ and $\mathcal{F} = \text{RF}_{\text{Dom}, \text{Rng}}$ be random oracles. Let A be some pro-adversary against C_2^f . Let S be an OTM with an interface S_f that on $(d+n)$ -bit inputs returns n -bit strings. We utilize a simple game-playing argument in conjunction with a hybrid argument to bound the indistinguishability of C_2 by that of C_1 (with respect to simulator S). Figure 2 displays two games, game G0 (includes boxed statement) and game G1 (boxed statement removed). The first game G0 exactly simulates the oracles C_2^f and f . The second game G1 exactly simulates the oracles C_1^f and f . We thus have that $\Pr[A^{C_2^f, f} \Rightarrow 1] = \Pr[A^{\text{G0}} \Rightarrow 1]$ and $\Pr[A^{C_1^f, f} \Rightarrow 1] = \Pr[A^{\text{G1}} \Rightarrow 1]$. Since G0 and G1 are identical-until-bad we have by the fundamental lemma of game playing [21] that $\Pr[A^{\text{G0}} \Rightarrow 1] -$

<p><u>Prefix-free MD:</u> $\text{PRE}[c] = c^+(IV, \text{padPF}(\cdot))$ where $\text{padPF}: \{0, 1\}^* \rightarrow D^+$ is a prefix-free padding function</p>	<p><u>NMAC Transform:</u> $\text{NT}[c, g] = g(c^+(IV, \text{pad}(\cdot)))$ where $g: \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a function</p>
<p><u>Chop Solution:</u> $\text{CHP}[c] =$ first $n - s$ bits of $c^+(IV, \text{pad}(\cdot))$</p>	<p><u>HMAC Transform:</u> $\text{HT}[c] =$ $c(c^+(IV, 0^d \parallel \text{pad}(\cdot)) \parallel 0^{d-n} \parallel IV)$</p>

Fig. 3. The four MD variants proposed in [1] that are PRO-Pr but not CR-Pr

$\Pr[A^{G^1} \Rightarrow 1] \leq \Pr[A^{G^1} \text{ sets } bad]$. The right hand side is equal to 2^{-n} because f is a random oracle. Thus,

$$\begin{aligned}
 \mathbf{Adv}_{C_2, S}^{\text{pro}}(A_2) &= \Pr[A^{G^0} \Rightarrow 1] - \Pr[A^{G^1} \Rightarrow 1] + \\
 &\quad \Pr[A^{G^1} \Rightarrow 1] - \Pr[A^{\mathcal{F}, S^{\mathcal{F}}} \Rightarrow 1] \\
 &\leq \Pr[A^{G^1} \text{ sets } bad] + \Pr[A^{C_1^f, f} \Rightarrow 1] - \Pr[A^{\mathcal{F}, S^{\mathcal{F}}} \Rightarrow 1] \\
 &= \frac{1}{2^n} + \mathbf{Adv}_{C_1, S}^{\text{pro}}(A_1).
 \end{aligned}$$

□

4.2 Insecurity of Proposed PRO-Pr Transforms

COLLISION-RESISTANCE PRESERVATION. The result above tells us that PRO-Pr does not imply CR-Pr for arbitrary schemes. What about MD variants? One might hope that the mechanisms used to create a PRO-Pr MD variant are sufficient for rendering the variant CR-Pr also. This is not true. In fact *all* previously proposed MD variants proven to be PRO-Pr *are not* CR-Pr. The four variants are summarized in Fig. 3 and below, see [1] for more details.

The first transform is *Prefix-free MD* specified by $\text{PRE}[c] = c^+(IV, \text{padPF}(\cdot))$. It applies a prefix-free padding function padPF to an input message and then uses the MD iteration. The padding function padPF must output strings that are a multiple of d bits with the property that for any two strings $M \neq M'$, $\text{padPF}(M)$ is not a prefix of $\text{padPF}(M')$. The *Chop solution* simply drops s bits from the output of the MD iteration applied to a message. The *NMAC transform* applies a second, distinct compression function to the output of an MD iteration; it is defined by $\text{NT}[c, g] = g(c^+(IV, \text{pad}(\cdot)))$, where g is a function from n bits to n bits (distinct from h). Lastly, the *HMAC Transform* is defined by $\text{HT}[c] = c(c^+(IV, 0^d \parallel \text{pad}(\cdot)) \parallel 0^{d-n} \parallel IV)$. This transform similarly utilizes enveloping: the MD iteration is fed into c in a way that distinguishes this last call from the uses of c inside the MD iteration. The prepending of a d -bit string of zeros to an input message helps ensure that the envelope acts differently than the first compression function application.

Let $IV = 0^n$. We shall use the collision-resistant hash function h that maps 0^{d+n} to 0^n (defined in Sect. 4.1). We first show that the PRE construction, while being PRO-Pr for all prefix-free encodings, is not CR-Pr for all prefix-free encodings. Let $\text{padPF}(M) = g_2(M)$ from Sect. 3.3 of [1]. Briefly, $g_2(M) = 0 \parallel M_1, \dots, 0 \parallel M_{k-1}, 1 \parallel M_k$ for $M_1 \parallel \dots \parallel M_k \stackrel{d-1}{\leftarrow} M \parallel 10^r$, where $r = (d-1) - ((|M|+1) \bmod d-1)$. (That is we append a one to M , and then enough zero's to make a string with length a multiple of $d-1$.) Now let $X = 0^{d-1}$ and $Y = 0^{2(d-1)}$. Then we have that $\text{PRE}^h(X) = \text{PRE}^h(Y)$ and no collisions against h occur. We should note that *some* prefix-free encodings will render PRE CR-Pr, for example any that also include strengthening. The important point here is that strengthening does not ensure prefix-freeness and vice-versa.

For the other three constructions, we assume that $\text{pad}(M)$ simply appends a one and then enough zeros to make a string with length a multiple of d . Let $X = 0^d$ and $Y = 0^{2d}$. Then we have that $\text{CHP}^h(X) = \text{CHP}^h(Y)$ and $\text{NT}^h(X) = \text{NT}^h(Y)$ and $\text{HT}^h(X) = \text{HT}^h(Y)$. Never is there a collision generated against h .

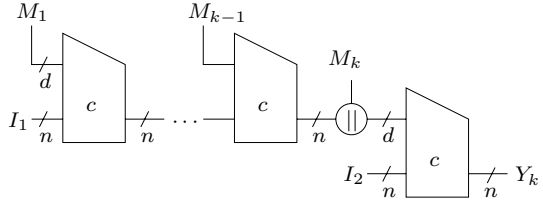
The straightforward counter-examples exploit the weakness of the basic MD transform. As noted previously, the MD transform does not give any guarantees about collision resistance, and only when we consider particular padding functions (i.e., `pad64`) can we create a CR-Pr transform. Likewise, we have illustrated that the mechanisms of prefix-free encodings, dropping output bits, and enveloping do nothing to help ensure collision-resistance is preserved, even though they render the transforms PRO-Pr. To properly ensure preservation of both properties, we must specify transforms that make use of mechanisms that ensure collision-resistance preservation *and* mechanisms that ensure pseudorandom oracle preservation. In fact, it is likely that adding strengthening to these transforms would render them CR-Pr. However, as we show in the next section, our new construction (with strengthening) is already more efficient than these constructions (without strengthening).

PRF PRESERVATION. It is not formally meaningful to compare PRF preservation with PRO preservation, since the resulting schemes in either case are different types of objects (one keyed and one keyless). However we can look at particular transforms. Of the four proposed by Coron et al. only PRE is known to be PRF-Pr. Let e be a PRF. Since we are using the key-via-IV strategy, the keyed version of PRE^e is $\text{PRE}_K^e(M) = e^+(K, \text{padPF}(M))$. This is already known to be a good PRF [13]. As for the other three transforms, it is unclear whether any of them are PRF-Pr. For NT, we note that the security will depend greatly on the assumptions made about g . If g is a separately keyed PRF, then we can apply the proof of NMAC [4]. On the other hand, if g is not one-way, then an adversary can determine the values produced by the underlying MD iteration and mount simple length-extension attacks. Instead of analyzing these transforms further (which are not CR-Pr anyway), we look at a new construction.

5 The EMD Transform

We propose a transform that is CR-Pr, PRO-Pr, and PRF-Pr. Let n, d be numbers such that $d \geq n + 64$. Let $c: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$ be a function and let $D^\circ = \cup_{i \geq 1} \{0, 1\}^{(i+1)d-n}$. Then we define the enveloped Merkle-Damgård iteration $c^\circ: \{0, 1\}^{2n} \times D^\circ \rightarrow \{0, 1\}^n$ on c by the algorithm given below.

Algorithm $c^\circ(I_1, I_2, M)$:
 $M_1 \cdots M_k \stackrel{d}{\leftarrow} M$
 $P \leftarrow M_1 \cdots M_{k-1}$
ret $c(c^+(I_1, P) \parallel M_k \parallel I_2)$



To specify our transform we require a padding function $\text{padEMD}: \{0, 1\}^{\leq 2^{64}} \rightarrow D^\circ$ for which the last 64 bits of $\text{padEMD}(M)$ encodes $|M|$. Fix $IV1, IV2 \in \{0, 1\}^n$ with $IV1 \neq IV2$. Then we specify the *enveloped Merkle-Damgård transform* $\text{EMD}[c] = c^\circ(IV1, IV2, \text{padEMD}(\cdot))$.

EMD utilizes two main mechanisms for ensuring property preservation. The first is the well-known technique of strengthening: we require a padding function that returns a string appended with the 64-bit encoding of the length. This ensures that EMD preserves collision-resistance. The second technique consists of using an ‘extra’ compression function application to envelope the internal MD iteration. It is like the enveloping mechanism used by Maurer and Sjoden in a different setting [18] (which is discussed in more detail in the full version of the paper [19]), but distinct from prior enveloping techniques used in the current setting. First, it includes message bits in the envelope’s input (in NMAC/HMAC and HT, these bits would be a fixed constant, out of adversarial control). This results in a performance improvement since in practice it is always desirable to have d as large as possible relative to n (e.g., in SHA-1 $d = 512$ and $n = 160$). Second, it utilizes a distinct initialization vector to provide (with high probability) domain separation between the envelope and internal applications of the compression function. This mechanism allows us to avoid having to use other previously proposed domain separation techniques while still yielding a PRO-Pr transform. (The previous techniques were prefix-free encodings or the prepending of 0^d to messages, as used in the HT transform; both are more costly.)

5.1 Security of EMD

COLLISION-RESISTANCE PRESERVATION. Let $h: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$ be a collision resistant hash function. Then any adversary which finds collisions against EMD^h (two messages $M \neq M'$ for which $\text{EMD}^h(M) = \text{EMD}^h(M')$) will necessarily find collisions against h . This can be proven using a slightly modified version of the proof that SMD is collision-resistant [2,3], and we therefore omit the details. The important intuition here is that embedding the length of messages in the last block is crucial; without the strengthening the scheme would

not be collision resistant (similar attacks as those given in Section 4 would be possible).

PRO PRESERVATION. Now we show that EMD is PRO-Pr. We first prove a slightly different transform is PRO-Pr and then show that EMD reduces to this other transform. Let $f, g = \text{RF}_{d+n,n}$ be random oracles. For any strings $P_1 \in D^+$ and $P_2 \in \{0, 1\}^{d-n}$ we define the function gf^+ : $D^\circ \rightarrow \{0, 1\}^n$ by $gf^+(P \parallel S) = g(f^+(IV1, P_1) \parallel P_2 \parallel IV2)$. This function is essentially EMD^f , except that we replace the envelope with an independent random oracle g . The following lemma states that gf^+ is a pseudorandom oracle.

Lemma 1. [gf^+ is a PRO] *Let $f, g = \text{RF}_{d+n,n}$. Let A be an adversary that asks at most q_L left queries, q_f right f -queries, q_g right g -queries and runs in time t . Then*

$$\text{Adv}_{gf^+, SB}^{\text{pro}}(A) \leq \frac{(q_L + q_g)^2 + q_f^2 + q_g q_f}{2^n}$$

where $SB = (SB_f, SB_g)$ is defined in Fig. 4 and $q_{SB} \leq q_g$ and $t_{SB} = \mathcal{O}(q_f^2 + q_g q_f)$.

We might hope that this result is given by Theorem 4 from [1], which states that $\text{NT}^{f,g}$ is indifferentiable from a random oracle. Unfortunately, their theorem statement does not allow for adversarially-specified bits included in the input to g . Thus we give a full proof of Lemma 1, found in the full version of the paper [19]. The next theorem captures the main result, and its proof is also in the full version. For completeness, we provide the simulators $SB = (SB_f, SB_g)$ and SA in Fig. 4.

Theorem 1. [EMD is PRO-Pr] *Fix n, d , and let $IV1, IV2 \in \{0, 1\}^n$ with $IV1 \neq IV2$. Let $f = \text{RF}_{d+n,n}$ and $\mathcal{F} = \text{RF}_{*,n}$ be random oracles. Let A be an adversary that asks at most q_L left queries (each of length no larger than ld bits), q_1 right queries with lowest n bits not equal to $IV2$, q_2 right queries with lowest n bits equal to $IV2$, and runs in time t . Then*

$$\text{Adv}_{\text{EMD}, SA}^{\text{pro}}(A) \leq \frac{(q_L + q_2)^2 + q_1^2 + q_2 q_1}{2^n} + \frac{lq_L^2}{2^n}.$$

where the simulator SA is defined in Fig. 4 and $q_{SA} \leq q_2$ and $t_{SA} = \mathcal{O}(q_1^2 + q_2 q_1)$.

PRF PRESERVATION. We utilize the key-via-IV strategy to create a keyed version of our transform, which is $\text{EMD}_{K_1, K_2}^e(M) = e^\circ(K_1, K_2, M)$ (for some PRF e). The resulting scheme is very similar to NMAC, which we know to be PRF-Pr [5]. Because our transform allows direct adversarial control over a portion of the input to the envelope function, we can not directly utilize the proof of NMAC (which assumes instead that these bits are fixed constants). However, the majority of the proof of NMAC is captured by two lemmas, The first (Lemma 3.1 [5]) shows (informally) that the keyed MD iteration is unlikely to have outputs that collide. The second lemma (Lemma 3.2 [5]) shows that composing the keyed MD iteration with a separately keyed PRF yields a PRF. We omit the details.

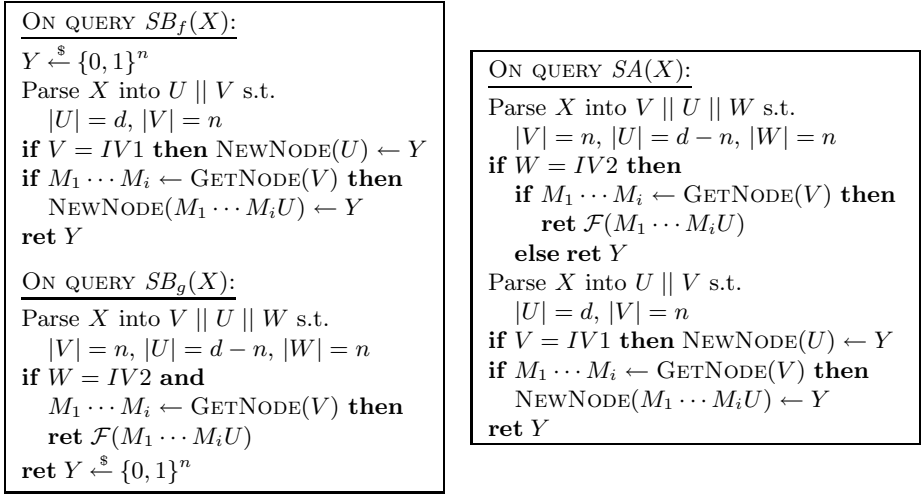


Fig. 4. Pseudocode for simulators SB (Lemma 1) and SA (Theorem 1)

Theorem 2. [EMD is PRF-Pr] Fix n, d and let $e: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$ be a function family keyed via the low n bits of its input. Let A be a prf-adversary against keyed EMD using q queries of length at most m blocks and running in time t . Then there exists prf-adversaries A_1 and A_2 against e such that

$$\text{Adv}_{\text{EMD}_{K_1, K_2}}^{\text{prf}}(A) \leq \text{Adv}_e^{\text{prf}}(A_1) + \binom{q}{2} \left[2m \cdot \text{Adv}_e^{\text{prf}}(A_2) + \frac{1}{2^n} \right]$$

where A_1 utilizes q queries and runs in time at most t and A_2 utilizes at most two oracle queries and runs in time $\mathcal{O}(mT_e)$ where T_e is the time for one computation of e .

Acknowledgments

The authors are supported in part by NSF grant CNS 0524765 and a gift from Intel Corporation.

References

1. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Advances in Cryptology - CRYPTO '05. Volume 3621 of Lecture Notes in Computer Science, Springer (2004) 21–39.
2. Merkle, R.C.: One way hash functions and DES. In: Advances in Cryptology - CRYPTO '89. Volume 435 of Lecture Notes in Computer Science, Springer (1989) 428–446.
3. Damgård, I.: A design principle for hash functions. In: Advances in Cryptology - CRYPTO '89. Volume 435 of Lecture Notes in Computer Science, Springer (1989) 416–427.

4. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: *Advances in Cryptology - CRYPTO '96*. Volume 1109 of *Lecture Notes in Computer Science*, Springer (1996) 1–15.
5. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: *Advances in Cryptology - CRYPTO '06*. Volume 4117 of *Lecture Notes in Computer Science*, Springer (2006) 602–619.
6. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: *CCS '93*, ACM Press (1993) 62–73.
7. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: *Advances in Cryptology - EUROCRYPT '94*. Volume 950 of *Lecture Notes in Computer Science*, Springer (1994) 92–111.
8. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: *Advances in Cryptology - EUROCRYPT '96*. Volume 1070 of *Lecture Notes in Computer Science*, Springer (1996) 399–416.
9. RSA Laboratories: RSA PKCS #1 v2.1: RSA Cryptography Standards (2002).
10. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: *TCC '04*. Volume 2951 of *Lecture Notes in Computer Science*, Springer (2004) 21–39.
11. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. *J. ACM* **51**(4) (2004) 557–594.
12. Bellare, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In Cachin, C., Camenisch, J., eds.: *Advances in Cryptology - EUROCRYPT '04*. Volume 3027 of *Lecture Notes in Computer Science*, Springer (2004) 171–188.
13. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: the cascade construction and its concrete security. In: *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society (1996) 514–523.
14. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: *Advances in Cryptology - CRYPTO '05*. Volume 3621 of *Lecture Notes in Computer Science*, Springer (2005) 17–36.
15. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: *Advances in Cryptology - EUROCRYPT '05*. Volume 3494 of *Lecture Notes in Computer Science*, Springer (2005) 19–35.
16. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: message authentication under weakened assumptions. In: *Advances in Cryptology - CRYPTO '99*. Volume 1666 of *Lecture Notes in Computer Science*, Springer (1999) 252–269.
17. Bellare, M., Rogaway, P.: Collision-Resistant Hashing: Towards Making UOWHF's Practical. In: *Advances in Cryptology - CRYPTO '97*. Volume 1294 of *Lecture Notes in Computer Science*, Springer (1997) 470–484.
18. Maurer, U., Sjödin, J.: Single-key AIL-MACs from any FIL-MAC. In: *ICALP '05*. Volume 3580 of *Lecture Notes in Computer Science*, Springer (2005) 472–484.
19. Bellare, M., Ristenpart, T.: Multi-property-preserving Hash Domain Extension and the EMD Transform (full version of this paper) (2006) <http://www.cse.ucsd.edu/users/mihir>.
20. National Institute of Standards and Technology: FIPS PUB 180-1: Secure Hash Standard. (1995) Supersedes FIPS PUB 180 1993 May 11.
21. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: *Advances in Cryptology - EUROCRYPT '06*. Volume 4004 of *Lecture Notes in Computer Science*, Springer (2006) 409–426.

Combining Compression Functions and Block Cipher-Based Hash Functions

Thomas Peyrin¹, Henri Gilbert¹, Frédéric Muller², and Matt Robshaw¹

¹France Télécom R&D, Issy les Moulineaux, France

²HSBC, Paris, France

{thomas.peyrin, henri.gilbert, matt.robshaw}@orange-ft.com,
frederic.muller@hsbc.fr

Abstract. The design of secure compression functions is of vital importance to hash function development. In this paper we consider the problem of combining smaller trusted compression functions to build a larger compression function. This work leads directly to impossibility results on a range of block cipher-based hash function constructions.

Keywords: block ciphers, compression functions, hash functions.

1 Introduction

Cryptographic hash functions are an important tool in cryptography. Informally, a cryptographic hash function H takes an input of variable size and returns a hash value of fixed length while satisfying the properties of preimage resistance, second preimage resistance, and collision resistance [26]. For a secure hash function that gives an n -bit output, compromising these properties should require 2^n , 2^n , and $2^{n/2}$ operations respectively.

The pioneering work of Merkle and Damgård [7,27] showed how to construct a secure hash function from a *compression function* h that has a fixed-length input, consisting of a *chaining variable* and a message extract, and gives a fixed-length output. A variety of interesting results [8,12,13] have provided a greater understanding of the Merkle-Damgård approach to the serial application of such a compression function.

Generally speaking, there are two popular approaches to building a compression function for use in a cryptographic hash function. The first is to use a compression function of a dedicated design while the second is to build a compression function around an established, and trusted, block cipher. While most widely-deployed hash functions [30,37] use a compression function of dedicated design, recent attacks [39,40] have demonstrated that there is much to learn. Instead, there is now much renewed interest in using a block cipher as the basis for a compression function.

It might be argued that the compression functions of common dedicated hash functions such as MD5 [37] and SHA-1 [30] are built on block ciphers; by removing the feed-forward from compression functions in the MD-family we are

left with a reversible component that can be used as a block cipher (such as SHACAL [9] in the case of SHA-1). But these block ciphers cannot be afforded the same level of trust as the leading standardised block ciphers [29,31], and instead block cipher-based hash functions are traditionally viewed as techniques to build a secure compression function from a trusted and standardised cipher. Much progress on using block ciphers in this way has already been made. Black *et al* [2] built on the work of Preneel [32] to present a range of secure $2n$ - to n -bit compression functions built around an n -bit block cipher that takes an n -bit key. Among these are the well-known Davies-Meyer, Matyas-Meyer-Oseas, and Miyaguchi-Preneel constructions. We therefore have many secure compression functions in hand whose chaining variable is the same size as the block size. However, a hash function built on a compression function with n bits of output can only offer a security level of at most $2^{n/2}$ operations. Since a security level of 2^{128} bits is often desired, we need to construct compression functions with outputs of at least 256 bits, a requirement that cannot be immediately met by the standardised block ciphers in hand.

Our difficulties begin, therefore, when we try to build secure compression functions whose output size is greater than the block size of the underlying block cipher. This is not a new problem and there has been mixed success in constructing $2n$ -bit hash functions from an n -bit block cipher [4,5,14,19,21,33,35]. While limitations have been identified in many constructions [14], Hirose [10] has demonstrated the security of a family of double block-length hash functions using two independent block ciphers with key length twice the block length. This is a property shared by AES-256 [29] and IDEA [20] among others with a particular instance of this construction being the long-standing ABREAST-DM [19].

While the case of block ciphers provided the initial motivation for our work, our results are essentially about compression functions. In this paper we explore the problem of combining compression functions that we know to be secure. These smaller compression functions can be of any type—dedicated, number theoretic, block cipher-based—and our aim is to build a secure compression function with a longer chaining variable. Thus the results are broader than block cipher-based hashing, though this is where there is an immediate, practical, and at times surprising, impact. The paper is organised as follows. In Section 2 we establish the framework and we make some initial observations in Section 3. After discussing some generic attacks in Section 4, we derive criteria for combining compression functions in Section 5 and demonstrate a range of impossibility results and potential constructions in Section 6. We then draw our conclusions and highlight opportunities for future work.

2 Notation and Model

In this paper we consider building larger compression functions from smaller trusted ones. We will assume that the underlying secure compression functions have k inputs of n bits and that the output is n bits in length. Details on the construction of secure compression functions will not be important to our results.

However, in the specific case of a block cipher with equal key and block size we have $k = 2$, while for a key size twice the block size we have $k = 3$. We could also use a compression function based on a tweaked block cipher [3,23] or a dedicated design (if we were willing to claim their security as secure compression functions) and we might then have $k > 3$ depending on the sizes of the chaining variable and message input. This flexible approach was pursued by Knudsen and Preneel in a series of papers [16,17,18].

This work is not a proof oriented paper, so we follow [18]: a collision resistant hash function or compression function outputting n bits is called *ideal* if the best algorithm to find a collision is a brute-force collision search; such an attack requires on average $\Theta(2^{n/2})$ evaluations of the hash function. Similarly, a preimage (resp. 2^{nd} -preimage) resistant hash function or compression function with n -bit output is called *ideal* if the best algorithm to find a preimage (resp. 2^{nd} -preimage) is a brute-force preimage (resp. 2^{nd} -preimage) search; such an attack requires on average $\Theta(2^n)$ evaluations of the hash function.

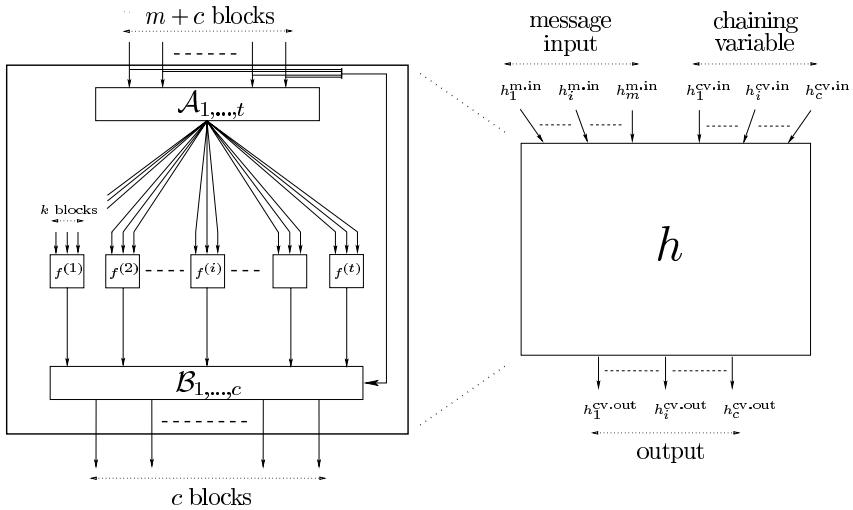


Fig. 1. The compression function h built from t compression functions $f^{(i)}$ each taking k inputs of n bits and delivering an n -bit output. m stands for message and cv for chaining variable.

In our constructions we will use t ideal n -bit compression functions to construct a secure compression function h that compresses $(m + c)n$ bits to cn bits. One important aspect to what follows in this paper is that we require the t internal ideal compression functions to act independently. Exactly how these are instantiated is outside the scope of this paper, but it is an important issue in practice. It is, however, an issue that has been addressed before and, under the assumption that the underlying block cipher is good, we can enforce independence of the fundamental compression functions by fixing bits of the underlying

“keys” to distinct values [18] or by using constants [11] to diversify the “keys” used in the compression function.

We will describe the inputs (resp. outputs) to the internal component compression functions as *internal inputs* (resp. *internal outputs*). These are distinguished from the *external inputs* and *external outputs* to the larger compression function h that we are trying to build. The $m + c$ inputs to h , each of n bits, will be denoted by $h_1^{m.in}, \dots, h_m^{m.in}, h_1^{cv.in}, \dots, h_c^{cv.in}$ and we denote the c n -bit output blocks by $h_1^{cv.out}, \dots, h_c^{cv.out}$.

The internal inputs will be derived as a linear combination of the external inputs to h , and we will derive the output from h as a linear combination of the internal outputs from the t ideal compression functions. Thus, the kt inputs to the internal compression functions $f_j^{(i)}$ ($1 \leq i \leq t$ and $1 \leq j \leq k$) will be linear functions of the external inputs and for each compression function $f^{(i)}$ we have

$$\begin{pmatrix} f_1^{(i)} \\ \vdots \\ f_k^{(i)} \end{pmatrix} = \mathcal{A}_i \cdot (h_1^{m.in}, \dots, h_m^{m.in}, h_1^{cv.in}, \dots, h_c^{cv.in})^T.$$

where \mathcal{A}_i is a $(k \cdot n \times (m + c) \cdot n)$ binary matrix, consisting of $(n \times n)$ blocks which are either zero or the identity matrix, corresponding to the compression function $f^{(i)}$. Taken together, such matrices define a mixing layer among the inputs to the t compression functions and we call this the *input layer*. Similarly, the external outputs from h are any linear combination of the t compression function outputs. This is the *output layer* and for the external outputs $h_i^{cv.out}$ ($1 \leq i \leq c$) we have

$$\begin{pmatrix} h_1^{cv.out} \\ \vdots \\ h_c^{cv.out} \end{pmatrix} = \mathcal{B}(f_{out}^{(1)}, \dots, f_{out}^{(t)}).$$

where \mathcal{B} is a $(c \cdot n \times t \cdot n)$ binary matrix, consisting of $(n \times n)$ blocks which are either zero or the identity matrix. This is illustrated in Figure 1. Note that we allow the possibility of a feedforward of the external inputs *around* the compression functions. We actually ignore this feature in the remainder of the paper, since we observe that incorporating a feedforward according to Figure 1 does not help prevent the attacks we consider in this paper.

We also recall the established fact [19,25] that

“...applying any simple (in both directions) invertible transformation to the input and to the output of the hash round function yields a new hash round function with the same security as the original one.”

We accept that such invertible transformations may well be applied to the external inputs and outputs of h before the input layer and after the output layer. But since they can have no cryptanalytic effect we ignore them.

Finally, we emphasize that we have restricted ourselves to parallel constructions where we compute $f_{out}^{(i)}$ as a linear combination of the external inputs. This

is a natural limitation that encompasses most previously established schemes and offers obvious performance benefits in hardware implementation. We also note that the structural observations of Joux [12], Dean [8], and Kelsey and Schneier [13], do not relate to the task of building a larger compression function from a layer of parallel compression functions, but only to the usual Merkle-Damgård iteration of the final compression function that results.

3 First Observations

Our model for combining compression functions is both natural and powerful. To illustrate we might consider some of the more prominent block cipher-based compression functions, and Appendix A shows how the compression function of MDC-2 fits our framework with parameters $c = 2$, $t = 2$, $k = 2$, and $m = 1$ (the two internal compression functions being Matyas-Meyer-Oseas constructions), while the schemes proposed by Nandi *et al.* [28] have (c, t, k, m) parameter sets $(2, 3, 2, 1)$ and $(2, 3, 3, 2)$. Other schemes with appropriate parameters are provided below.

Name		c	t	k	m	<i>Cryptanalysis</i>
MDC-2	[5]	2	2	2	1	[32]
PBGV	[33]	2	2	2	2	[19]
ABREAST-DM	[19]	2	2	3	1	-
PARALLEL-DM	[21]	2	2	2	2	[14]
Hirose family	[10]	2	2	3	1	-
Nandi <i>et al.</i> N_1	[28]	2	3	2	1	[15]
Nandi <i>et al.</i> N_2	[28]	2	3	3	2	[15]

Like other compression function-based work, we cover instances where the underlying block cipher has different block and key lengths. However, unlike many previous constructions, we consider using t internal compression functions to derive c blocks of output with $t \geq c$. This allows us to make a fundamental distinction between previous work and that presented in this paper.

We identify the size of the output chaining variable that is required, and hence the number of output blocks c . Then, by considering established attacks, we achieve bounds on t that give us the minimum number of compression functions required to achieve the desired security level. We achieve this by a suitable analysis of the *output layer*. Our goal is to derive schemes that offer an optimal level of security of 2^{nc} work effort for preimage attacks and $2^{\frac{nc}{2}}$ for collision attacks. This nicely complements the work of Knudsen and Preneel [16,17,18], where the security of potentially non-optimal constructions is analysed via consideration of the *input layer*.

First, we observe the following series of implications. Given a set of parameters (c, t, k, m) for some construction, we use $(c, t, k, m) \in S$ to denote that a construction with ideal collision resistance with these parameters exists and we use $(c, t, k, m) \notin S$ to denote the fact that no such scheme can exist for this parameter set.

Implications 1. *Given c, t, k , and m all ≥ 1 , we have the following four sets of pairwise equivalent implications:*

$$\begin{array}{ll}
 (c, t, k, m) \notin S \Rightarrow (c, t, k, m + 1) \notin S & (c, t, k, m + 1) \in S \Rightarrow (c, t, k, m) \in S \\
 (c, t, k, m) \in S \Rightarrow (c, t, k + 1, m) \in S & (c, t, k + 1, m) \notin S \Rightarrow (c, t, k, m) \notin S \\
 (c, t, k, m) \in S \Rightarrow (c, t + 1, k, m) \in S & (c, t + 1, k, m) \notin S \Rightarrow (c, t, k, m) \notin S \\
 (c, t, k, m) \notin S \Rightarrow (c + 1, t, k, m) \notin S & (c + 1, t, k, m) \in S \Rightarrow (c, t, k, m) \in S.
 \end{array}$$

Justification: Suppose that there exists a secure design with parameter set (c, t, k, m) . If we replace one message block by a constant then we still have a secure scheme. Thus the first implications are true. If we can use one additional input for every inner compression function, then we can use them so that none has any influence over the output. Thus, the second set of implications are true. If we have an additional compression function, we can still build a secure scheme by simply ignoring it. Thus the third set of implications is true. The final implications reflect the natural conjecture that constructing an ideal compression function of output size $c + 1$ blocks is harder than constructing an ideal compression function of output size c blocks. \square

The above implications are simple but useful. For MDC-2 the corresponding parameter set is $(2, 2, 2, 1)$; a double block-length construction using two compression functions, each taking two equal-sized inputs (key and message) and processing one message block at each iteration. As shown in Section 4, $(2, 2, 2, 1) \notin S$. Yet, there has been much effort in building schemes with a better rate, *i.e.* hashing more than one message at each iteration, for which one corresponding parameter set would be $(2, 2, 2, 2)$. But we have that $(2, 2, 2, 1) \notin S \Rightarrow (2, 2, 2, 2) \notin S$ and such efforts cannot succeed¹.

4 Generic Attacks

In this section we consider two attacks that have been used in the literature. By generalising these attacks we are able to make statements about the impossibility of certain constructions. More importantly, we extract criteria for the successful design of a compression function with an intended level of security.

4.1 Attack Method: DF

The first generic attack depends on what we term the number of *degrees of freedom*. It resembles the classic divide-and-conquer strategy from other cryptanalytic fields and can be applied to many proposals. The idea is to isolate, and attack, a linear combination of the output blocks but to keep at least one external input block free from conditions. Then, the free input can be determined separately at the end of the attack. Attacks on MDC-2 provide a good example [32]

¹ To avoid any confusion we emphasize that the double block-length construction of Hirose [10] has parameter set $(2, 3, 2, 1)$ since it uses a block cipher with a key that is twice the block size.

and an equivalent representation of MDC-2 is provided in Figure 3. To find a preimage, one can attack the two branches independently. Finding a preimage for one branch will fix two inputs to the overall compression function and since we have three external inputs M , H_1 , and H_2 there remains one external input free, *i.e.* one degree of freedom. Thus, we can independently use the free input to obtain a preimage for the other branch by brute force. The attack has work effort proportional to 2^n operations instead of the intended 2^{2n} . A collision attack works in a corresponding way. Consideration of this attack gives some of the bounds in [16,17,18]. We use it again here.

4.2 Attack Method: MUL

The second attack uses *multi-collisions* and *multi-preimages* and is described in [36,15]. Similar considerations were used in a different way in [18]. For the attack to be successful, the compression function must satisfy several structural conditions. First, the attacker identifies a linear combination Z of the external outputs of h that depends on a non-empty set G_Z of compression functions $\{f^{(i)}\}$. Next, the attacker identifies two external input blocks X and Y . The external input X should influence the internal inputs to a subset G_X of the compression functions in G_Z . Similarly the external input block Y should influence the internal inputs to a subset, G_Y , of the compression functions in G_Z . It is important to identify X and Y (and hence G_X and G_Y) so that $G_X \cap G_Y = \emptyset$.

We now describe the attack in terms of finding preimages. The attacker fixes values to all the external input blocks except the previously identified inputs X and Y . Then, each value of X (resp. Y) is used to generate an internal output value for each $f^{(i)}$ in G_X (resp. G_Y). Thus, the attacker effectively compiles two lists L_X and L_Y each containing 2^n elements where, for every possible value of X and Y , all the internal outputs of the set of $\{f^{(i)}\}$ in G_X and G_Y are stored. Using Wagner's technique [38] these two lists can be joined in 2^n operations to obtain a third list L_Z that contains all (X, Y) (with $X \in L_X$ and $Y \in L_Y$) yielding the target image for the external output block. Since L_X and L_Y both have almost 2^n elements, we expect L_Z to contain almost 2^n elements.

At this stage we have found 2^n preimages to one external output block at a cost proportional to 2^n operations. If h has c output blocks, then an entry in the list L_z will give a good preimage for *all* c external output blocks with a probability of $2^{-(c-1)n}$. Thus, we repeat this procedure for $2^{(c-2)n}$ allocations of the $m + c - 2$ input variables distinct from x and y in order to find a valid preimage with a probability close to 1. The attack requires $2^{(c-1)n}$ operations instead of $2^{c \cdot n}$ in the ideal case. The collision attack works in a similar fashion.

5 Security Criteria

The compression function h that we wish to build takes $m + c$ external input blocks and each internal compression function $f^{(i)}$ takes k internal input blocks, defined by input matrices \mathcal{A}_i . Since we can apply any invertible transformation to the inputs of h , the important criteria for the input layer is the dimension

of the vector space generated by columns of the matrices \mathcal{A}_i . This is already explored in existing work [14]. Considering the results in Section 4, we can make the following observations.

- To prevent attack DF, every external output block $h_i^{\text{cv.out}}$ must depend on all external input blocks $h_1^{\text{m.in}}, \dots, h_m^{\text{m.in}}, h_1^{\text{cv.in}}, \dots, h_c^{\text{cv.in}}$ no matter which invertible transformations of the external inputs and outputs are used.
- We say that an *identified pair* of external input blocks is a pair (A, B) where A and B both appear within the internal inputs to some $f^{(i)}$. (For example, with $f^{(i)}(A, B \oplus C)$, the identified pairs (A, B) , (A, C) , and (B, C) appear in $f^{(i)}$.) Then, in order to prevent attack MUL, *every possible pair* of external input blocks must appear as an identified pair for every invertible combination of external output blocks $h_i^{\text{cv.out}}$. This applies, no matter which invertible transformations of the external inputs and outputs are used.

We now consider the secure combination of independent compression functions.

5.1 Deriving Valid Parameter Sets

Rather than using the identified attacks and their generalisations to break specific proposals, we use them to derive general lower bounds on the number of smaller ideal compression functions needed to derive a larger ideal compression function. More precisely, for a set of k -input secure compression functions, *i.e.* compressing kn to n bits, we ascertain the minimum number t_{\min} of compression functions required to build a secure compression function producing cn bits, since they must resist DF and MUL attacks. To do this, we adopt a two-phase approach. First we establish a bound d on the number of compression functions we require when considering any single linear combination of the c output blocks. We then derive a bound t_{\min} on the minimum number of compression functions that are required when simultaneously considering all c output blocks in the chaining variable (see Table 1).

Initial bounds on d . First, we consider attack DF and we observe that since each compression function takes k input blocks, and that there are $m+c$ external input blocks to h , then we must have at least $\lceil \frac{m+c}{k} \rceil$ compression functions. Thus, every external output block depends on at least $\lceil \frac{m+c}{k} \rceil$ internal output blocks. This is required for every linear combination of the external outputs and so we have $d \geq \lceil \frac{m+c}{k} \rceil$.

Improved bounds on d . By considering attack DF we can derive the basic bounds on d given above. However a generic analysis allows us to improve on this bound by ensuring that a proposed configuration of compression functions also resists attack MUL. While the style of analysis is generic and can be reused for different parameter sets, it is most easily described by reference to one particular instance.

Suppose that we consider the parameter set given by $m+c=3$ and $k=2$ with A, B , and C denoting the three n -bit inputs to the compression function. Our basic bound gives $d \geq 2$, so here we assume that $d=2$. Suppose that an

Table 1. The minimum number t_{\min} of compression functions required to resist DF and MUL attacks, for parameter set (c, t_{\min}, k, m)

Parameters			Basic Bounds		Improved	
c	k	m	d	t_{\min}	d	t_{\min}
2	2	1	2	3	3	5
2	2	2	2	3	3	5
2	3	1	1	2	-	-
2	3	2	2	3	3	5
3	2	1	2	4	3	6
3	2	2	3	6	4	7
3	3	1	2	4	3	6
3	3	2	2	4	3	6
4	2	1	3	7	4	8
4	2	2	3	7	4	8
4	3	1	2	5	3	7
4	3	2	2	5	3	7

external output block $h_i^{\text{cv.out}}$, or more generally a linear combination Z of one or more output blocks, is bound to only two compression functions f_1 and f_2 . Then we have that $Z = f_1(X_1, X_2) \oplus f_2(X_3, X_4)$ where X_1, X_2, X_3 , and X_4 are linear combinations of A, B , and C .

The rank of the vector space $\langle X_1, X_2, X_3, X_4 \rangle$ spanned by X_1, \dots, X_4 must be equal to three since otherwise attack DF would apply. Therefore, one can extract from $\langle X_1, X_2, X_3, X_4 \rangle$ three elements which together form a basis of $\langle A, B, C \rangle$. Without loss of generality, we assume that $\langle X_1, X_2, X_3 \rangle = \langle A, B, C \rangle$ and there exist binary coefficients α_i so that $X_4 = \alpha_1 A \oplus \alpha_2 B \oplus \alpha_3 C$. We cannot have α_1 or α_2 equal to zero, since otherwise the pairs (A, C) and (B, C) would not be encountered in either f_1 or f_2 and the attack MUL would apply. So we can assume without loss of generality, that $\alpha_1 = 1$ and $\alpha_2 = 1$. If we now apply the invertible change of variables $A' = A \oplus B, B' = B$, and $C' = C$, Z can be rewritten as $Z = f_1(A' \oplus B', B') \oplus f_2(C', A' \oplus \alpha_3 C')$. Since (B', C') is not encountered in either f_1 or f_2 , then the attack MUL applies. Thus $d \geq 3$. Note that such reasoning also applies when $m + c \geq 3$, thus if $m + c \geq 3$ and $k = 2$ we have $d \geq 3$.

This style of reasoning allows us to improve most of the bounds on d by considering the applicability of the second generic attack MUL. The sole exception is the parameter set $c = 2, k = 3$, and $m = 1$ which corresponds to the provably secure scheme of Hirose and will be discussed in Section 6.2.

Initial bounds on t . We now turn bounds on d into bounds on the minimum number of compression functions that must be used, t_{\min} . While any linear combination of the c external outputs must depend on at least d inner compression functions, a bound on the minimal number t_{\min} of compression functions is not immediate. Here we derive a value for t independently of the analysis needed to derive d .

In the simple case that $c = 2$ a combinatorial style of reasoning can be used and this shows that $t_{\min} \geq \frac{3d}{2}$ if d is even and $t_{\min} \geq \frac{3(d-1)}{2} + 2$ otherwise. However a more flexible approach, scaling better to larger parameters, uses an analogy with coding theory.

Consider vectors of t elements (corresponding to the number of internal compression functions) and attach to each external output block $h_i^{c,\text{out}}$ a vector v_i whose value is determined by whether an internal compression function influences $h_i^{c,\text{out}}$. If compression function $f^{(j)}$ is active in $h_i^{c,\text{out}}$ then set the j^{th} entry of v_i to 1, otherwise it has the value 0. For example, if $t = 3$ and for some proposed construction only $f^{(1)}$ and $f^{(3)}$ are involved in $h_i^{c,\text{out}}$, then we set $v_i = (1, 0, 1)$.

In turning our result on d into a constraint on t_{\min} , we consider the problem of looking for a binary code of length t with minimal distance d and dimension c . The Singleton bound yields $c \leq t - d + 1$ and so $t \geq c + d - 1$. The Hamming bound is tighter, but is more involved and given in Appendix B.

Improved bounds on t . It is interesting to note that configurations with particular features might allow a dedicated, and potentially tighter, analysis for the bounds on t . An example is given in Appendix C. However since such analysis does not apply to the general model we have established, (it relies on a particular form to the input layer), we do not use it in the derivation of the bounds in Table 1.

6 Constructions

Given a set of parameters (c, t, k, m) it is easy to use the newly established bounds to check whether, according to our criteria, the scheme is necessarily insecure. Turning this around, if one wants to build a scheme with some pre-defined c , k , and m then one can compute a lower bound t_{\min} on the number of internal compression functions that must be used, in a parallel configuration that we consider in Figure 1.

6.1 Impossible Constructions

Using the bounds established in Section 5 we first consider interesting parameter sets such as $c \in \{2, 3, 4\}$, $k \in \{2, 3\}$, and $m \in \{1, 2\}$. These correspond to cases where we aim to obtain double, triple, or quadruple block-length constructions, using a block cipher with key size the same or twice the block size, and processing either one or two blocks of message.

We use the bounds on d and once c , k , and m are chosen we search for the smallest t that satisfy our bounds. We thus derive an integer t_{\min} for the minimum number of independent compression functions that must be used in the specified construction. Note that a given t_{\min} does not mean that secure schemes with t_{\min} inner compression functions necessarily exist. Rather, no secure scheme can exist with fewer independent compression functions of the stated type.

Immediately there are interesting results and we note that secure schemes with (c, t, k, m) parameters $(2, 3, 2, 1)$ or $(2, 3, 3, 2)$ are impossible. These correspond to the schemes of Nandi *et al.* [28]. Since our bounds are derived by generalising attacks on [28] we expect this to be the case. However, constructions using four inner compression functions, would still be insecure.

Indeed, for the most natural case with $c = 2, k = 2,$ and $m = 1,$ the case of DES and AES-128, one must use at least five inner compression functions in a parallel framework to obtain a secure hash function offering 64-bit and 128-bit security respectively. This is more than one might have expected. The case of a quadruple block-length output is even more dramatic. If one wished to design a compression function that used AES-128 as a building block but offered 256-bit security, then one would be required to use at least eight parallel instantiations of AES-128 to produce a secure compression function.

6.2 Proposed Constructions

Figure 2 shows a $(2, 5, 2, 1)$ -scheme that is secure against the attacks considered in this paper. Further research will determine whether other attacks apply. However, this scheme is one from a range of double block-length hash function

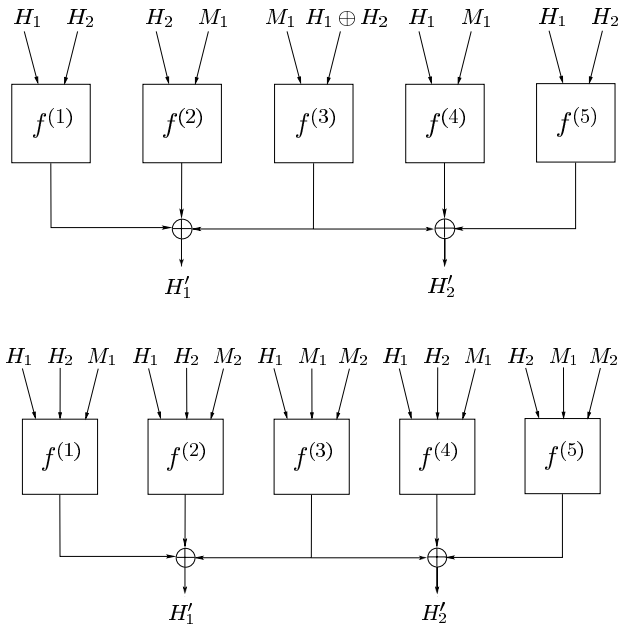


Fig. 2. A $(2, 5, 2, 1)$ and a $(2, 5, 3, 2)$ construction. For the first construction each (independent) inner compression function can be instantiated using a block cipher with equal key and block size. For the second construction, the key size is double the block size. M_1, M_2 are n -bit message blocks; H_1, H_2 are n -bit incoming chaining variable blocks and H'_1, H'_2 are n -bit output chaining variable blocks.

constructions that might be instantiated with AES-128 or other block ciphers with identical block and key sizes. Note that this is the only such construction that remains uncompromised. Figure 2 also depicts a $(2, 5, 3, 2)$ -construction that resists our generic attacks, meets our bound, and could be instantiated with AES-256 or a cipher like IDEA (or even TWO-KEY TRIPLE-DES) with a key length twice the block length. The parameter set $(2, 2, 3, 1)$ is covered by Hirose.

A particularly simple set of parameters satisfies $k \geq m + c$ when all external inputs can be accommodated within each internal compression function and $d = 1$. Thus, we derive a secure compression function with $t = c$ without requiring additional internal compression functions. We only need to ensure that all external input blocks are used directly in every internal compression function with any free internal inputs fixed to a constant value. Then every external output needs to be bound to one, and only one, internal compression function. Hirose [10] has already studied members of this family of block cipher based hash functions and proved their security in both the random oracle model and in the ideal cipher model when the compression functions are instantiated using a Davies-Meyer construction.

7 Conclusions

In this paper we have analyzed techniques to construct a larger compression function by combining smaller, trusted, compression functions. By generalising attacks in the literature, we are able to establish conditions on the type and number of components that are required to ensure that the constructions are not vulnerable to a range of powerful and general attacks.

This work has a direct and immediate application to the construction of block cipher-based hash functions for which the length of the hash output is greater than the block size of the underlying block cipher. The most important conclusion to draw is that it is actually rather difficult to use multiple instantiations of a block cipher to build a secure compression function; or at least to do so in a particularly efficient way. For example, when using AES-128 for double block-length hashing, one must use at least five parallel instantiations of AES-128 to derive a compression function offering 128-bit security respectively. To achieve 256-bit security, one must use eight. This is a surprisingly high number of block cipher calls, particularly so when we consider that this is merely to avoid the application of generic attacks.

While there are many possible generalisations to the framework used in this paper, we have provided a natural and broad framework for the analysis of schemes of this type. Extensions to this work, including identifying schemes that achieve the most efficient permissible bounds, is the subject of ongoing research.

Acknowledgements

The authors would like to thank Sébastien Kunz-Jacques, Yannick Seurin, and the program committee for their valuable comments.

References

1. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62-73. 1993.
2. J. Black, P. Rogaway, and T. Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320-335. Springer-Verlag, 2002.
3. J. Black, M. Cochran, and T. Shrimpton. On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 526-541. Springer-Verlag, 2005.
4. L. Brown, J. Pieprzyk, and J. Seberry. LOKI - a Cryptographic Primitive for Authentication and Secrecy Applications. In J. Pieprzyk and J. Seberry, editors, *Advances in Cryptology – AUSCRYPT '90*, volume 453 of *Lecture Notes in Computer Science*, pages 229-236. Springer-Verlag, 1990.
5. D. Coppersmith, S. Pilpel, C.H. Meyer, S.M. Matyas, M.M. Hyden, J. Oseas, B. Brachtel, and M. Schilling. Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function. U.S. Patent No. 4,908,861, March 13, 1990.
6. J-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430-448. Springer-Verlag, 2005.
7. I. Damgård. A Design Principle for Hash Functions. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416-427. Springer-Verlag, 1989.
8. R.D. Dean. *Formal Aspects of Mobile Code Security*. PhD thesis, Princeton University, 1999.
9. H. Handschuh, L.R. Knudsen, and M.J.B. Robshaw. Analysis of SHA-1 in Encryption Mode. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 70-83. Springer-Verlag, 2001.
10. S. Hirose. Provably Secure Double-block-length Hash Functions in a Black-box Model. In C. Park and S. Chee, editors, *Information Security and Cryptology – ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 330-342. Springer-Verlag, 2004.
11. S. Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In M.J.B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*.
12. A. Joux. Multi-collisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306-316. Springer-Verlag, 2004.
13. J. Kelsey and B. Schneier. Second Preimages on n -bit Hash Functions for Much Less Than 2^n Work. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474-490. Springer-Verlag, 2005.

14. L.R. Knudsen and X. Lai. New Attacks on All Double Block Length Hash Functions of Hash Rate 1, Including the Parallel-DM. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 410–418. Springer-Verlag, 1994.
15. L.R. Knudsen and F. Muller. Some Attacks Against a Double Length Hash Proposal. In B. Roy, editor, *Advances in Cryptology – ASIACRYPT '05*, volume 3788 of *Lecture Notes in Computer Science*, pages 462–473. Springer-Verlag, 2005.
16. L.R. Knudsen and B. Preneel. Hash Functions Based on Block Ciphers and Quaternary Codes. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology – ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 77–90. Springer-Verlag, 1996.
17. L.R. Knudsen and B. Preneel. Fast and Secure Hashing Based on Codes. In B.S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 485–498. Springer-Verlag, 1997.
18. L.R. Knudsen and B. Preneel. Construction of Secure and Fast Hash Functions Using Nonbinary Error-Correcting Codes. *IEEE Transactions on Information Theory*, 48(9):2524–2539, 2002.
19. X. Lai and J.L. Massey. Hash Functions Based on Block Ciphers. In R. A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer-Verlag, 1992.
20. X. Lai, J.L. Massey, and S. Murphy. Markov Ciphers and Differential Cryptanalysis. In D.W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer-Verlag, 1991.
21. X. Lai, C. Waldvogel, W. Hohl, and T. Meier. Security of Iterated Hash Functions Based on Block Ciphers. In D.R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 379–390. Springer-Verlag, 1993.
22. S. Lucks. A Failure-Friendly Design Principle for Hash Functions. In B. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer-Verlag, 2005.
23. M. Liskov, R.L. Rivest, and D. Wagner. Tweakable Block Ciphers. In M. Yung, editor, *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, 2002.
24. R. Matsumoto, K. Kurosawa, and T. Itoh. Primal-Dual Distance Bounds of Linear Codes with Application to Cryptography. IACR Cryptology ePrint Archive, Report 2005/194. Available from: <http://eprint.iacr.org>.
25. W. Meier and O. Staffelbach. Nonlinearity Criteria for Cryptographic Functions. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 549–562. Springer-Verlag, 1989.
26. A.J. Menezes, S.A. Vanstone, and P.C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.
27. R.C. Merkle. One Way Hash Functions and DES. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1989.
28. M. Nandi, W. Lee, K. Sakurai, and S. Lee. Security Analysis of a 2/3-rate Double Length Compression Function in Black-box Model. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption – FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 243–254. Springer-Verlag, 2005.
29. National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, November 2001 . Available from: <http://csrc.nist.gov>.

30. National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard, August 2002 . Available from: <http://csrc.nist.gov>.
31. National Institute of Standards and Technology. SP800-67: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, May 2004 . Available from: <http://csrc.nist.gov>.
32. B. Preneel. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
33. B. Preneel, A. Bosselaers, R. Govaerts, and J. Vandewalle. Collision-free Hash Functions Based on Block Cipher Algorithms. In *Proceedings 1989 International Carnahan Conference on Security Technology (Oct 3-5 1989: Zurich, Switzerland)*, pages 203–210. IEEE, 1989. IEEE catalog number 89CH2774-8.
34. B. Preneel, R. Govaerts, and J. Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In D.R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer-Verlag, 1993.
35. J.-J. Quisquater and M. Girault. 2n-bit Hash-functions Using n-bit Symmetric Block Cipher Algorithms. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 102–109. Springer-Verlag, 1989.
36. B. Preneel, R. Govaerts, and J. Vandewalle. On the Power of Memory in the Design of Collision Resistant Hash Functions. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology – ASIACRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 105-121. Springer-Verlag, 1992.
37. Ronald L. Rivest. RFC 1321: The MD5 Message-Digest Algorithm, April 1992 . Available from: <http://www.ietf.org/rfc/rfc1321.txt>.
38. D. Wagner. A Generalized Birthday Problem. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer-Verlag, 2002.
39. X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer-Verlag, 2005.
40. X. Wang, Y.L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer-Verlag, 2005.

Appendix A: Some Established Constructions

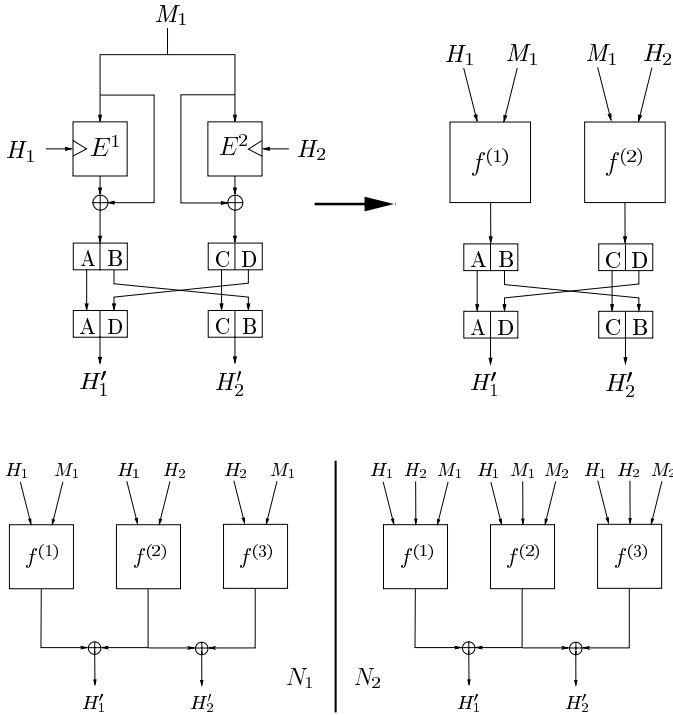


Fig. 3. Mapping the compression functions of MDC-2 and Nandi *et al.* to our framework. Recall that simple invertible transformations such as a swap can be ignored [19]. M_1, M_2 are n -bit message blocks; H_1, H_2 are n -bit incoming chaining variable blocks and H'_1, H'_2 are n -bit output chaining variable blocks.

Appendix B: The Hamming Bound

While it is more difficult to exploit, the Hamming bound is tighter than the Singleton bound. Here we give an improved version of the Hamming bound [24]:

$$\begin{cases} c \leq t - \log_2 \left(\sum_{i=0}^{\frac{d-1}{2}} \binom{t}{i} \right) & \text{if } d \text{ is odd, and} \\ c \leq t - \log_2 \left(\binom{t-1}{\frac{d}{2}-1} + \sum_{i=0}^{\frac{d}{2}-1} \binom{t}{i} \right) & \text{if } d \text{ is even.} \end{cases}$$

This can be used to give a bound on t in terms of c and d . The table below allows us to compare the Singleton and the Hamming bound for some parameter sets used in Table 1.

<i>Parameters</i>		<i>Bounds</i>	
<i>c</i>	<i>d</i>	Singleton	Hamming
2	1	2	2
2	2	3	3
2	3	4	5
3	2	4	4
3	3	5	6
3	4	6	7
4	2	5	5
4	3	6	7
4	4	7	8

Appendix C: Preferred Bounds on t in a Restricted Model

While the bounds derived in this appendix do not apply to the general model, it is interesting to see what can be achieved with some minor restrictions to the general framework. Here we consider the impact of a simplified input layer and we assume that each of the kt internal inputs is one of the $m + c$ external inputs. This is far more restrictive than the general case of a linear combination of the external inputs and so it is not surprising that we can derive better bounds.

From the previous analysis we know that every possible pair of external inputs must be present in at least one of the internal compression functions involved in any linear combination of the external output blocks. We have $N_C = (m + c) \cdot (m + c - 1)/2$ different pairs. In each internal compression function, we can have at most $N_K = k \cdot (k - 1)/2$ pairs present. Each of the N_C pairs must appear in at least c different internal compression functions since otherwise there would exist a linear combination of the external outputs which would involve none of these internal compression functions and attack MUL would apply. We thus have:

$$t \geq \frac{c \cdot (m + c) \cdot (m + c - 1)}{k \cdot (k - 1)}.$$

This reasoning can also be applied to attack DF since we have at most $m + c$ different vectors as input to the internal compression functions. Each external input block must appear in at least c different internal compression functions, otherwise some linear combinations of the external outputs would not depend on this external input block. We can put k blocks in one inner function and thus we have:

$$t \geq \frac{c \cdot (m + c)}{k}.$$

These bounds are often much better than the general case and illustrate the importance of the input layer. A weak input layer can dramatically increase the minimum number of compression functions required for a secure construction.

A Scalable Password-Based Group Key Exchange Protocol in the Standard Model

Michel Abdalla and David Pointcheval

Departement d'Informatique École normale supérieure, CNRS
{Michel.Abdalla, David.Pointcheval}@ens.fr
<http://www.di.ens.fr/~{mabdalla,pointache}>

Abstract. This paper presents a secure constant-round password-based group key exchange protocol in the common reference string model. Our protocol is based on the group key exchange protocol by Burmester and Desmedt and on the 2-party password-based authenticated protocols by Gennaro and Lindell, and by Katz, Ostrovsky, and Yung. The proof of security is in the standard model and based on the notion of smooth projective hash functions. As a result, it can be instantiated under various computational assumptions, such as decisional Diffie-Hellman, quadratic residuosity, and N -residuosity.

Keywords: Smooth Projective Hash Functions, Password-based Authentication, Group Key Exchange.

1 Introduction

Key exchange is one of the most useful tools in public-key cryptography, allowing users to establish a common secret which they can then use in applications to achieve both privacy and authenticity. Among the examples of key exchange protocols, the most classical one is the Diffie-Hellman protocol [22]. Unfortunately, the latter only works between two players and does not provide any authentication of the players.

Group Key Exchange. Group key exchange protocols are designed to provide a pool of players communicating over an open network with a shared secret key which may later be used to achieve cryptographic goals like multicast message confidentiality or multicast data integrity. Secure virtual conferences involving up to one hundred participants is an example.

Due to the usefulness of group key exchange protocols, several papers have attempted to extend the basic Diffie-Hellman protocol to the group setting. Nonetheless, most of these attempts were rather informal or quite inefficient in practice for large groups. To make the analyses of such protocols more formal, Bresson et al. [11,16] introduced a formal security model for group key exchange protocols, in the same vein as [6,7,4]. Moreover, they also proposed new protocols, referred to as group Diffie-Hellman protocols, using a ring structure for the communication, in which each player has to wait for the message from his

predecessor before producing his own. Unfortunately, the nature of their communication structure makes their protocols quite impractical for large groups since the number of rounds of communication is linear in the number of players.

A more efficient and practical approach to the group key exchange problem is the one proposed by Burmester and Desmedt [17,18], in which they provide a *constant-round* Diffie-Hellman variant. Their protocol is both scalable and efficient, even for large groups, since it only requires 2 rounds of broadcasts. Thus, with reasonable time-out values, one could always quickly decide whether or not a protocol has been successfully executed. Furthermore, their protocol has also been formally analyzed, in the above security framework [30].

Password-Based Authenticated Key Exchange. The most classical way to add authentication to key exchange protocols is to sign critical message flows. In fact, as shown by Katz and Yung [30] in the context of group key exchange protocols, this technique can be made quite general and efficient, converting any scheme that is secure against passive adversaries into one that is secure against active ones. Unfortunately, such techniques require the use of complex infrastructures to handle public keys and certificates. One way to avoid such infrastructures is to use passwords for authentication. In the latter case, the pool of players who wants to agree on a common secret key only needs to share a low-entropy password—a 4-digit pin-code, for example—against which an exhaustive search is quite easy to perform. In password-based protocols, it is clear that an outsider attacker can always guess a password and attempt to run the protocol. In case of failure, he can try again with a different guess. After each failure, the adversary can erase one password. Such an attack, known as “on-line exhaustive search” cannot be avoided, but the damage it may cause can be mitigated by other means such as limiting the number of failed login attempts. A more dangerous threat is the “off-line exhaustive search”, also known as “dictionary attack”. It would mean that after one failure, or even after a simple eavesdropping, the adversary can significantly reduce the number of password candidates.

In the two-party case, perhaps the most well known Diffie-Hellman variant is the encrypted key exchange protocol by Bellare and Merritt [8]. However, its security analyses [4,10,13,14] require ideal models, such as the random oracle model [5] or the ideal cipher model. The first practical password-based key exchange protocol, without random oracles, was proposed by Katz et al. [28] in the common reference string model and it is based on the Cramer-Shoup cryptosystem [19]. Their work was later extended by Gennaro and Lindell [24] using the more general smooth projective hash function primitive [19,20,21].

In the group key exchange case, very few protocols have been proposed with password authentication. In [12,15], Bresson et al. showed how to adapt their group Diffie-Hellman protocols to the password-based scenario. However, as the original protocols on which they are based, their security analyses require ideal models and the total number of rounds is linear in the number of players, making their schemes impractical for large groups. More recently, several constant-round password-based group key exchange protocols have been proposed in the literature by Abdalla et al. [1], by Dutta and Barua [23], and by Kim, Lee, and

Lee [31]. All of these constructions are based on the Burmester and Desmedt protocol [17,18] and are quite efficient, but their security analyses usually require the random oracle and/or the ideal cipher models.¹ Independently of and concurrently to our work, a new constant-round password-based group key exchange protocol has been proposed by Bohli et al. [9]. Their protocol is more efficient than ours and also enjoys a security proof in the standard model.

Contributions. In this paper, we propose the first password-based authenticated group key exchange protocol in the standard model. To achieve this goal, we extend the Gennaro-Lindell framework [24] to the group setting, using ideas similar to those used in the Burmester-Desmedt protocol [17,18]. In doing so, we take advantage of the smooth projective hash function primitive [20] to avoid the use of ideal models. Our protocol has several advantages. First, it is efficient both in terms of communication, only requiring 5 rounds, and in terms of computation, with a per-user computational load that is linear in the size of the group. Second, like the Burmester-Desmedt protocol, our protocol is also contributory since each member contributes equally to the generation of the common session key. Such property, as pointed out by Steiner, Tsudik and Waidner [33], may be essential for certain distributed applications. Finally, as in the Gennaro-Lindell framework [24], our protocol works in the common reference string model and is quite general, being built in a modular way from four cryptographic primitives: a labeled encryption scheme secure against chosen-ciphertext attacks, a signature scheme, a family of smooth projective hash functions, and a family of universal hash functions. Thus, it can be instantiated under various computational assumptions, such as decisional Diffie-Hellman, quadratic residuosity, and N -residuosity (see [24]). In particular, the Diffie-Hellman variant (based on the Cramer-Shoup cryptosystem [19]) can be seen as a generalization of the KOY protocol [28] to the group setting.

2 Security Model

The security model for password-based group key exchange protocols that we present here is the one by Bresson et al. [15], which is based on the model by Bellare et al. [4] for 2-party password-based key exchange protocols.

Protocol participants. Let \mathcal{U} denote the set of potential participants in a password-based group key exchange protocol. Each participant $U \in \mathcal{U}$ may belong to several subgroups $\mathcal{G} \subseteq \mathcal{U}$, each of which has a unique password $\text{pw}_{\mathcal{G}}$ associated to it. The password $\text{pw}_{\mathcal{G}}$ of a subgroup \mathcal{G} is known to all the users $U_i \in \mathcal{G}$.

Protocol execution. The interaction between an adversary \mathcal{A} and the protocol participants only occurs via oracle queries, which model the adversary capabilities in a real attack. During the execution of the protocol, the adversary may

¹ In fact, in [1], Abdalla et al. showed that the protocols by Dutta and Barua [23] and by Kim, Lee, and Lee are insecure by presenting concrete attacks against these schemes.

create several instances of a participant and several instances of the same participant may be active at any given time. Let $U^{(i)}$ denote the instance i of a participant U and let b be a bit chosen uniformly at random. The query types available to the adversary are as follows:

- $Execute(U_1^{(i_1)}, \dots, U_n^{(i_n)})$: This query models passive attacks in which the attacker eavesdrops on honest executions among the participant instances $U_1^{(i_1)}, \dots, U_n^{(i_n)}$. It returns the messages that were exchanged during an honest execution of the protocol.
- $Send(U^{(i)}, m)$: This query models an active attack, in which the adversary may tamper with the message being sent over the public channel. It returns the message that the participant instance $U^{(i)}$ would generate upon receipt of message m .
- $Reveal(U^{(i)})$: This query models the misuse of session keys by a user. It returns the session key held by the instance $U^{(i)}$.
- $Test(U^{(i)})$: This query tries to capture the adversary’s ability to tell apart a real session key from a random one. It returns the session key for instance $U^{(i)}$ if $b = 1$ or a random key of the same size if $b = 0$.

Partnering. Following [30], we define the notion of partnering via session and partner identifiers. Let the session identifier sid^i of a participant instance $U^{(i)}$ be a function of all the messages sent and received by $U^{(i)}$ as specified by the group key exchange protocol. Let the partner identifier pid^i of a participant instance $U^{(i)}$ is the set of all participants with whom $U^{(i)}$ wishes to establish a common secret key. Two instances $U_1^{(i_1)}$ and $U_2^{(i_2)}$ are said to be partnered if and only if $pid_1^{i_1} = pid_2^{i_2}$ and $sid_1^{i_1} = sid_2^{i_2}$.

Freshness. Differently from [30], our definition of freshness does not take into account forward security as the latter is out of the scope of the present paper. Let acc^i be true if an instance $U^{(i)}$ goes into an accept state after receiving the last expected protocol message and false otherwise. We say that an instance $U^{(i)}$ is fresh if $acc^i = true$ and no *Reveal* has been asked to $U^{(i)}$ or to any of its partners.

Correctness. For a protocol to be correct, it should always be the case that, whenever two instances $U_1^{(i_1)}$ and $U_2^{(i_2)}$ are partnered and have accepted, both instances should hold the same non-null session key.

Indistinguishability. Consider an execution of the group key exchange protocol P by an adversary \mathcal{A} , in which the latter is given access to the *Reveal*, *Execute*, *Send*, and *Test* oracles and asks a single *Test* query to a *fresh* instance, and outputs a guess bit b' . Let $SUCC$ denote the event b' correctly matches the value of the hidden bit b used by the *Test* oracle. The AKE-IND advantage of an adversary \mathcal{A} in violating the indistinguishability of the protocol P and the advantage function of the protocol P , when passwords are drawn

from a dictionary \mathcal{D} , are respectively $\text{Adv}_{P,\mathcal{D}}^{\text{ake-ind}}(\mathcal{A}) = 2 \cdot \Pr[\text{SUCC}] - 1$ and $\text{Adv}_{P,\mathcal{D}}^{\text{ake-ind}}(t, R) = \max_{\mathcal{A}}\{\text{Adv}_{P,\mathcal{D}}^{\text{ake-ind}}(\mathcal{A})\}$, where maximum is over all \mathcal{A} with time-complexity at most t and using resources at most R (such as the number of queries to its oracles). The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the experiments defining the security plus the code size.

We say that a password-based group key exchange protocol P is secure if the advantage of any polynomial-time adversary is only negligibly larger than $O(q/|\mathcal{D}|)$, where q is number of different protocol instances to which the adversary has asked *Send* queries. Given that the dictionary size can be quite small in practice, the hidden constant in the big- O notation should be as small as possible (preferably 1) for a higher level of security.

3 Building Blocks

3.1 Universal Hash Function Families

One of the tools used in our protocol is a family of universal hash functions. A family \mathcal{UH} of universal hash function is a map $\mathbf{K} \times \mathbf{G} \mapsto \mathbf{R}$, where \mathbf{K} is the key or seed space, \mathbf{G} is the domain of the hash function, and \mathbf{R} is the range. For each seed or key $k \in \mathbf{K}$, we can define a particular instance $\text{UH}_k : \mathbf{G} \mapsto \mathbf{R}$ of the family by fixing the key being used in the computation of the function. For simplicity, we sometimes omit the seed k from the notation when referring to a particular instance of the family. Let UH_k be a universal hash function chosen at random from a family \mathcal{UH} . One of the properties of universal hash function families in which we are interested is the one that says that, if an element g is chosen uniformly at random from \mathbf{G} , then the output distribution of $\text{UH}_k(g)$ is statistically close to uniform in \mathbf{R} [26].

3.2 Signatures

The signature scheme used in our protocol is the standard one introduced by Goldwasser, Micali, and Rivest [25]. A standard signature scheme $\text{SIG} = (\text{SKG}, \text{Sign}, \text{Ver})$ is composed of three algorithms. The key generation algorithm SKG takes as input 1^k , where k is a security parameter, and returns a pair (sk, vk) containing the secret signing key and the public verification key. The signing algorithm Sign takes as input the secret key sk and a message m and returns a signature σ for that message. The verification algorithm Ver on input (vk, m, σ) returns 1 if σ is a valid signature for the message m with respect to the verification key vk .

The security notion for signature schemes needed in our proofs is strong existential unforgeability under chosen-message attacks [25]. More precisely, let (sk, vk) be a pair of secret and public keys for a signature scheme SIG , let $\text{SIGN}(\cdot)$ be a signing oracle which returns $\sigma = \text{Sign}(sk, m)$ on input m , and let \mathcal{F} be an adversary. Then, consider the experiment in which the adversary \mathcal{F} , who is given access to the public key vk and to the signing oracle $\text{SIGN}(\cdot)$, outputs a pair

(m, σ) . Let $\{(m_i, \sigma_i)\}$ denote the set of queries made to the signing oracle with the respective responses and let SUCC denote the event in which $\text{Ver}(vk, m', \sigma') = 1$ and that $(m', \sigma') \notin \{(m_i, \sigma_i)\}$. The $\text{SIG-SUF-CMA-advantage}$ of an adversary \mathcal{F} in violating the chosen message security of the signature scheme SIG is defined as $\text{Adv}_{\text{SIG}, \mathcal{F}}^{\text{sig-suf-cma}}(k) = \Pr[\text{SUCC}]$. A signature scheme SIG is said to be $\text{SIG-SUF-CMA-secure}$ if this advantage is a negligible function in k for all polynomial time adversaries (PTAs) \mathcal{F} asking a polynomial number of queries to their signing oracle.

3.3 Labeled Encryption

The notion of labeled encryption, first formalized in the ISO 18033-2 standard [32], is a variation of the usual encryption notion that takes into account the presence of labels in the encryption and decryption algorithms. More precisely, in a labeled encryption scheme, both the encryption and decryption algorithms have an additional input parameter, referred to as a label, and the decryption algorithm should only correctly decrypt a ciphertext if its input label matches the label used to create that ciphertext.

Formally, a labeled encryption scheme $\text{LPKE} = (\text{LKG}, \text{Enc}, \text{Dec})$ consists of three algorithms. Via $(pk, sk) \xleftarrow{\$} \text{LKG}(1^k)$, where $k \in \mathbb{N}$ is a security parameter, the randomized key-generation algorithm produces the public and secret keys of the scheme. Via $c \xleftarrow{\$} \text{Enc}(pk, l, m; r)$, the randomized encryption algorithm produces a ciphertext c for a label l and message m using r as the randomness. Via $m \leftarrow \text{Dec}(sk, l, c)$, the decryption algorithm decrypts the ciphertext c using l as the label to get back a message m .

The security notion for labeled encryption is similar to that of standard encryption schemes. The main difference is that, whenever the adversary wishes to ask a query to his Left-or-Right encryption oracle, in addition to providing a pair of messages (m_0, m_1) , he also has to provide a target label l in order to obtain the challenge ciphertext c . Moreover, when chosen-ciphertext security (LPKE-IND-CCA) is concerned, the adversary is also allowed to query his decryption oracle on any pair (l, c) as long as the ciphertext c does not match the output of a query to his Left-or-Right encryption oracle whose input includes the label l . As shown by Bellare et al. in the case of standard encryption schemes [3], one can easily show that the Left-or-Right security notion for labeled encryption follows from the more standard Find-Then-Guess security notion (in which the adversary is only allowed a single query to his challenging encryption oracle).

3.4 Smooth Projective Hash Functions

The notion of projective hash function families was first introduced by Cramer and Shoup [20] as a means to design chosen-ciphertext secure encryption schemes. Later, Gennaro and Lindell [24] showed how to use such families to build secure password-based authenticated key exchange protocols. One of the properties that makes these functions particularly interesting is that, for certain points of their domain, their values can be computed by using either a *secret* hashing key or a

public projective key. While the computation using *secret* hashing key works for all the points in the domain of the hash function, the computation using a public *projective* key only works for a specified subset of the domain. A projective hash function family is said to be smooth if the value of the function on inputs that are outside the particular subset of the domain are independent of the projective key. In [24], the notion of smooth hash functions was presented in the context of families of hard (partitioned) subset membership problems. Here we follow the same approach.

HARD PARTITIONED SUBSET MEMBERSHIP PROBLEMS. Let $k \in \mathbb{N}$ be a security parameter. In a family of hard (partitioned) subset membership problem, we first specify two sets $\mathbf{X}(k)$ and $\mathbf{L}(k)$ in $\{0, 1\}^{\text{poly}(k)}$ such that $\mathbf{L}(k) \subseteq \mathbf{X}(k)$ as well as two distributions $D(\mathbf{L}(k))$ and $D(\mathbf{X}(k) \setminus \mathbf{L}(k))$ over $\mathbf{L}(k)$ and $\mathbf{X}(k) \setminus \mathbf{L}(k)$ respectively. Next, we specify a witness set $\mathbf{W}(k) \subseteq \{0, 1\}^{\text{poly}(k)}$ and a NP-relation $\mathbf{R}(k) \subseteq \mathbf{X}(k) \times \mathbf{W}(k)$ such that $x \in \mathbf{L}(k)$ if and only if there exists a witness $w \in \mathbf{W}(k)$ such that $(x, w) \in \mathbf{R}(k)$. Then, we say that a family of subset membership problems is hard if $(\mathbf{X}(k), \mathbf{L}(k), D(\mathbf{L}(k)), D(\mathbf{X}(k) \setminus \mathbf{L}(k)), \mathbf{W}(k), \mathbf{R}(k))$ instances can be efficiently generated, that a member element $x \in \mathbf{L}(k)$ can be efficiently sampled according to $D(\mathbf{L}(k))$ along with a witness $w \in \mathbf{W}(k)$ to the fact that $(x, w) \in \mathbf{R}(k)$, that non-member elements $x \in \mathbf{X}(k) \setminus \mathbf{L}(k)$ can be efficiently sampled according to $D(\mathbf{X}(k) \setminus \mathbf{L}(k))$, and that the distributions of member and non-member elements cannot be efficiently distinguished. The definition of hard *partitioned* subset membership problem is an extension of the one given above in which the set $\mathbf{X}(k)$ is partitioned in disjoint subsets $\mathbf{X}(k, i)$ for some index i and for which for all i it remains hard to distinguish an element $x \in \mathbf{L}(k, i)$ chosen according to a distribution $D(\mathbf{L}(k, i))$ from an element $x \in \mathbf{X}(k, i) \setminus \mathbf{L}(k, i)$ chosen according to a distribution $D(\mathbf{X}(k, i) \setminus \mathbf{L}(k, i))$.

HARD PARTITIONED SUBSET MEMBERSHIP PROBLEMS FROM LABELED ENCRYPTION. The families of hard partitioned subset membership problems in which we are interested are those based on LPKE-IND-CCA-secure labeled encryption schemes. More precisely, let $\mathcal{LPKE} = (\text{LKG}, \text{Enc}, \text{Dec})$ be a LPKE-IND-CCA-secure labeled encryption scheme and let pk be a public key outputted by the LKG algorithm for a given security parameter k . Let $\text{Enc}(pk)$ denote an efficiently recognizable superset of the space of all ciphertexts that may be outputted by the encryption algorithm Enc when the public key is pk and let \mathbf{L} and \mathbf{M} denote efficiently recognizable supersets of the label and message spaces. Using these sets, we can define a family of hard partitioned subset membership problems as follows. First, we define the sets \mathbf{X} and \mathbf{L} for the family of hard subset membership problems as $\mathbf{X}(pk) = \text{Enc}(pk) \times \mathbf{L} \times \mathbf{M}$ and $\mathbf{L}(pk) = \{(c, l, m) \mid \exists r \text{ s.t. } c = \text{Enc}(pk, l, m; r)\}$. Next, we define the partitioning of the sets \mathbf{X} and \mathbf{L} with respect to the message and label used in the encryption as $\mathbf{X}(pk, l, m) = \text{Enc}(pk) \times l \times m$ and $\mathbf{L}(pk, l, m) = \{(c, l, m) \mid \exists r \text{ s.t. } c = \text{Enc}(pk, l, m; r)\}$. The distribution $D(\mathbf{L}(pk, l, m))$ can then be defined by choosing a random $r \in \mathbf{R}$ and outputting the triple $(\text{Enc}(pk, l, m; r), l, m)$ with r as a witness. Likewise, the distribution $D(\mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m))$ can be defined by choosing a random $r \in \mathbf{R}$ and outputting the triple $(\text{Enc}(pk, l, m'; r), l, m)$, where m' is a dummy

message different from m but of the same length. Finally, we define the witness set $\mathbf{W}(pk)$ to be r and the NP-relation $\mathbf{R}(pk)$ in a natural way. It is easy to see that the hardness of distinguishing non-members from members follows from the LPKE-IND-CCA security of the labeled encryption scheme.

SMOOTH PROJECTIVE HASH FUNCTIONS. Let $\mathcal{HLPKE}(pk) = (\mathbf{X}(pk), \mathbf{L}(pk), D(\mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m)), D(\mathbf{L}(pk, l, m)), \mathbf{W}(pk), \mathbf{R}(pk))$ be a family of hard (partitioned) subset membership problems based on a LPKE-IND-CCA-secure labeled encryption scheme \mathcal{LPKE} with security parameter k . A family of smooth projective hash functions $\mathcal{HSH}(pk) = (\text{HashKG}, \text{ProjKG}, \text{Hash}, \text{ProjHash})$ associated with \mathcal{HLPKE} consists of four algorithms. Via $hk \xleftarrow{\$} \text{HashKG}(pk)$, the randomized key-generation algorithm produces hash keys $hk \in \mathbf{HK}(pk)$, where $k \in \mathbb{N}$ is a security parameter and pk is the public key of a labeled encryption scheme \mathcal{LPKE} . Via $phk \xleftarrow{\$} \text{ProjKG}(hk, l, c)$, the randomized key projection algorithm produces projected hash keys $phk \in \mathbf{PHK}(pk)$ for a hash key hk with respect to label l and ciphertext c . Via $g \leftarrow \text{Hash}(hk, c, l, m)$, the hashing algorithm computes the hash value $g \in \mathbf{G}(pk)$ of (c, l, m) using the hash key hk . Via $g \leftarrow \text{ProjHash}(phk, c, l, m; r)$, the projected hashing algorithm computes the hash value $g \in \mathbf{G}(pk)$ of (c, l, m) using the projected hash key phk and a witness r to the fact that c is a valid encryption of message m with respect to the public-key pk and label l .

PROPERTIES. The properties of smooth projective hash functions in which we are interested are correctness, smoothness, and pseudorandomness.

Correctness. Let \mathcal{LPKE} be a labeled encryption scheme and let pk be a public key outputted by the LKG algorithm for a given security parameter k . Let $c = \text{Enc}(pk, l, m; r)$ be the ciphertext for a message m with respect to public key pk and label l computed using r as the randomness. Then, for any hash key $hk \in \mathbf{HK}(pk)$ and projected hash key $phk \xleftarrow{\$} \text{ProjKG}(hk, l, c)$, the values $\text{Hash}(hk, c, l, m)$ and $\text{ProjHash}(phk, c, l, m, r)$ are the same.

Smoothness. Let $hk \in \mathbf{HK}(pk)$ be a hash key and let $phk \in \mathbf{PHK}(pk)$ be a projected hash key for hk with respect to l and c . Then, for every triple (c, l, m) for which c is *not* a valid encryption of message m with respect to the public-key pk and label l (i.e., $(c, l, m) \in \mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m)$), the hash value $g = \text{Hash}(hk, c, l, m)$ is statistically close to uniform in \mathbf{G} and independent of the values (phk, c, l, m) .

Pseudorandomness. Let \mathcal{LPKE} be a LPKE-IND-CCA-secure labeled encryption scheme, let pk be a public key outputted by the LKG algorithm for a given security parameter k , and let $(l, m) \in \mathbf{L} \times \mathbf{M}$ be a message-label pair. Then, for uniformly chosen hash key $hk \in \mathbf{HK}(pk)$ and randomness $r \in \mathbf{R}(pk)$, the distributions $\{c = \text{Enc}(pk, l, m; r), l, m, phk \xleftarrow{\$} \text{ProjKG}(hk, l, c), g \leftarrow \text{Hash}(hk, c, l, m)\}$ and $\{c = \text{Enc}(pk, l, m; r), l, m, phk \xleftarrow{\$} \text{ProjKG}(hk, l, c), g \xleftarrow{\$} \mathbf{G}\}$ are computationally indistinguishable.

EXAMPLES. To provide the reader with an idea of how efficient smooth projective hash functions are, we recall here the example given in [24] based on the Cramer-Shoup encryption scheme [19].

The labeled version of the Cramer-Shoup scheme works as follows. Let G be a cyclic group of prime order q where q is large. The key generation algorithm chooses two additional random generators g_1, g_2 in G , a collision-resistant hash function H , and random values $z, \tilde{z}_1, \tilde{z}_2, \hat{z}_1, \hat{z}_2$ in Z_q with $z \neq 0$. The secret key is set to $(z, \tilde{z}_1, \tilde{z}_2, \hat{z}_1, \hat{z}_2)$ and the public key is defined to be $(h, \tilde{h}, \hat{h}, g_1, g_2, H)$, where $h = g_1^z$, $\tilde{h} = g_1^{\tilde{z}_1} g_2^{\tilde{z}_2}$, and $\hat{h} = g_1^{\hat{z}_1} g_2^{\hat{z}_2}$. To encrypt a message $m \in G$ with respect to label l , the sender chooses $r \in Z_q$, and computes $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r \cdot m$, $\theta = H(l, u_1, u_2, e)$ and $v = (\tilde{h}\hat{h}^\theta)^r$. The ciphertext is $c = (u_1, u_2, e, v)$. To decrypt a ciphertext $c = (u_1, u_2, e, v)$ with respect to label l , the receiver computes $\theta = H(l, u_1, u_2, e)$ and tests if v equals $u_1^{\tilde{z}_1 + \theta \tilde{z}_1} u_2^{\tilde{z}_2 + \theta \hat{z}_2}$. If equality does not hold, it outputs \perp ; otherwise, it outputs $m = eu_1^{-z}$.

The smooth projective hashing for the labeled Cramer-Shoup encryption scheme is then defined as follows. The hash key generation algorithm **HashKG** simply sets the key hk to be the tuple (a_1, a_2, a_3, a_4) where each a_i is a random value in Z_q . The key projection function **ProjKG**, on input (hk, l, c) , first computes $\theta = H(l, u_1, u_2, e)$ and outputs $phk = g_1^{a_1} g_2^{a_2} h^{a_3} (\tilde{h}\hat{h}^\theta)^{a_4}$. The hash function **Hash** on input (hk, c, l, m) outputs $u_1^{a_1} u_2^{a_2} (e/m)^{a_3} v^{a_4}$. The projective hash function **ProjHash** on input (phk, c, l, m, r) simply outputs phk^r .

4 A Scalable Password-Based Group Key Exchange Protocol

In this section, we finally present our password-based group key exchange protocol. Our protocol is an extension of the Gennaro-Lindell password-based key exchange protocol [24] to the group setting and uses ideas similar to those used in the Burmester-Desmedt group key exchange protocol [18]. The Gennaro-Lindell protocol itself is an abstraction of the password-based key exchange protocol of Katz, Ostrovsky, and Yung [28,29]. Like the Gennaro-Lindell protocol, our protocol is built in a modular way from four cryptographic primitives: a LPKE-IND-CCA-secure labeled encryption scheme, a signature scheme, a family of smooth projective hash functions, and a family of universal hash functions. Thus, our protocol enjoys efficient instantiations based on the decisional Diffie-Hellman, quadratic residuosity, and N -residuosity assumptions (see [24]). Like the Burmester-Desmedt group key exchange protocol, our protocol only requires a constant number of rounds and low per-user computation.

As done in the Gennaro-Lindell protocol, we also assume the existence of a mechanism to allow parties involved in the protocol to differentiate between concurrent executions as well as identify the other parties with which they are interacting. As in their case, this requirement is only needed for the correct operation of the protocol. No security requirement is imposed on this mechanism.

4.1 Protocol Description

OVERVIEW. As in the Burmester-Desmedt protocol, our protocol assumes a ring structure for the users so that we can refer to the predecessor and successor of a user. Moreover, we associate each user with an index i between 1 and n , where n is the size of the group. After deciding on the order of the users, our protocol works as follows. First, each user in the group executes two correlated instances of the Gennaro-Lindell protocol, one with his predecessor and one with his successor so each user can authenticate his neighbors (this accounts for the first 3 rounds of the protocol). However, instead of generating a single session key in each of these instances, we modify the original Gennaro-Lindell protocol so that two independent session keys are generated in each session (this requires an extra hash key and an extra projection key per user). We then use the first one of these as a test key to authenticate the neighbor with whom that key is shared and we use the other one to help in the computation of the group session key, which is defined as the product of these latter keys. To do so, we add one more round of communication like in the Burmester-Desmedt protocol, so that each user computes and broadcasts the ratio of the session keys that he shares with his predecessor and successor. After this round, each user is capable of computing the group session key. However, to ensure that all users agree on the same key, a final round of signatures is added to the protocol to make sure that all users compute the group session key based on the same transcript. The key used to verify the signature of a user is the same one transmitted by that user in the first round of the Gennaro-Lindell protocol.

For a pictorial description of our protocol, please refer to Fig. 1. The formal description follows.

DESCRIPTION. Let $\mathcal{LPKE} = (\text{LKG}, \text{Enc}, \text{Dec})$ be a labeled encryption scheme, let $\text{SIG} = (\text{SKG}, \text{Sign}, \text{Ver})$ be a signature scheme, and let $\mathcal{HASH}(pk) = (\text{HashKG}, \text{ProjKG}, \text{Hash}, \text{ProjHash})$ be a family smooth projective hash functions based on \mathcal{LPKE} . Let $\text{UH} : \mathbf{G} \mapsto \{0, 1\}^{2l}$ and $\text{UH}' : \mathbf{G} \mapsto \{0, 1\}^l$ be two universal hash functions chosen uniformly at random from the families \mathcal{UH} and \mathcal{UH}' and let $\text{UH}_1(g)$ and $\text{UH}_2(g)$ refer to the first and second halves of $\text{UH}(g)$. Let U_1, \dots, U_n be the users wishing to establish a common secret key and let pw be their joint password chosen uniformly at random from a dictionary \mathbf{Dict} of size N . We assume pw either lies in the message space \mathbf{M} of \mathcal{LPKE} or can be easily mapped to it. Our protocol has a total of five rounds of communication and works as follows.

Initialization. A trusted server runs the key generation algorithm LKG on input 1^k , where $k \in \mathbb{N}$ is a security parameter, to obtain a pair (pk, sk) of secret and public keys and publishes the public key pk along with randomly selected universal hash function UH and UH' from the families \mathcal{UH} and \mathcal{UH}' .

Round 1. In this first round, each player U_i for $i = 1, \dots, n$ starts by setting the partner identifier pid_i to $\{U_1, \dots, U_n\}$. Then, each player U_i generates a pair (sk_i, vk_i) of secret and public keys for a signature scheme and a label $l_i = vk_i \parallel$

$U_1 \parallel \dots \parallel U_n$. Next, each player encrypts the joint group password \mathbf{pw} using the encryption algorithm \mathbf{Enc} with respect to the public key pk and label l_i using $r_i^{\mathbf{R}}$ as the randomness. Let $c_i^{\mathbf{R}}$ denote the resulting ciphertext (i.e., $c_i^{\mathbf{R}} = \mathbf{Enc}(pk, l_i, \mathbf{pw}; r_i^{\mathbf{R}})$). At the end of this round, each player U_i broadcasts the pair $(l_i, c_i^{\mathbf{R}})$.

Round 2. In this second round, each player U_i for $i = 1, \dots, n$ encrypts once more the joint group password \mathbf{pw} using the encryption algorithm \mathbf{Enc} with respect to the public key pk and label l_i using $r_i^{\mathbf{L}}$ as the randomness. Let $c_i^{\mathbf{L}}$ denote the resulting ciphertext (i.e., $c_i^{\mathbf{L}} = \mathbf{Enc}(pk, l_i, \mathbf{pw}; r_i^{\mathbf{L}})$). Next, each player U_i chooses a hash key $hk_i^{\mathbf{L}}$ uniformly at random from $\mathbf{HK}(pk)$ for the smooth projective hash function and then generates a projection key $phk_i^{\mathbf{L}}$ for it with respect to the pair $(c_{i-1}^{\mathbf{R}}, l_{i-1})$. That is, $phk_i^{\mathbf{L}} \stackrel{\$}{\leftarrow} \mathbf{ProjKG}(hk_i^{\mathbf{L}}, l_{i-1}, c_{i-1}^{\mathbf{R}})$. Here and in other parts of the protocol, the indices are taken modulo n . At the end of this round, each player U_i broadcasts the pair $(c_i^{\mathbf{L}}, phk_i^{\mathbf{L}})$.

Round 3. In this round, player U_i first chooses two new hash keys hk_i and $hk_i^{\mathbf{R}}$ uniformly at random from $\mathbf{HK}(pk)$ for the smooth projective hash function. Next, player U_i generates two projection keys phk_i and $phk_i^{\mathbf{R}}$ for the hash keys hk_i and $hk_i^{\mathbf{R}}$, both with respect to the pair $(c_{i+1}^{\mathbf{L}}, l_{i+1})$. That is, $phk_i \stackrel{\$}{\leftarrow} \mathbf{ProjKG}(hk_i, l_{i+1}, c_{i+1}^{\mathbf{L}})$ and $phk_i^{\mathbf{R}} \stackrel{\$}{\leftarrow} \mathbf{ProjKG}(hk_i^{\mathbf{R}}, l_{i+1}, c_{i+1}^{\mathbf{L}})$. Then, player U_i computes a test master key $X_i^{\mathbf{R}} = K_{i+1}^{\mathbf{L}} \cdot K_i^{\mathbf{R}}$ for its successor, where $K_i^{\mathbf{L}} \triangleq \mathbf{Hash}(hk_i^{\mathbf{L}}, c_{i-1}^{\mathbf{R}}, l_{i-1}, \mathbf{pw})$ and $K_i^{\mathbf{R}} \triangleq \mathbf{Hash}(hk_i^{\mathbf{R}}, c_{i+1}^{\mathbf{L}}, l_{i+1}, \mathbf{pw})$. Note that player U_i can compute $K_i^{\mathbf{R}}$ using $hk_i^{\mathbf{R}}$ and $K_{i+1}^{\mathbf{L}}$ using $phk_{i+1}^{\mathbf{L}}$ and the witness $r_i^{\mathbf{R}}$ to the fact that $c_i^{\mathbf{R}}$ is a valid encryption of \mathbf{pw} with respect to pk and l_i . Finally, player U_i computes a test key $test_i^{\mathbf{R}} = \mathbf{UH}_1(X_i^{\mathbf{R}})$, sets $T_i^{\mathbf{R}} = U_i \parallel U_{i+1} \parallel c_i^{\mathbf{R}} \parallel c_{i+1}^{\mathbf{L}} \parallel phk_i \parallel phk_i^{\mathbf{R}} \parallel phk_{i+1}^{\mathbf{L}}$ and computes a signature $\sigma_i^{\mathbf{R}}$ on $T_i^{\mathbf{R}}$ using sk_i . At the end of this round, player U_i broadcasts the tuple $(phk_i, phk_i^{\mathbf{R}}, test_i^{\mathbf{R}}, \sigma_i^{\mathbf{R}})$.

Round 4. In this round, each player U_i first verifies if the signature $\sigma_{i-1}^{\mathbf{R}}$ on the transcript $T_{i-1}^{\mathbf{R}}$ is correct using vk_{i-1} . If this check fails, then player U_i halts and sets $\mathbf{acc}_i = \mathbf{false}$. Otherwise, player U_i computes the values $K_i^{\mathbf{L}}$ and $K_{i-1}^{\mathbf{R}}$, using the hash key $hk_i^{\mathbf{L}}$ and the projection key $phk_{i-1}^{\mathbf{R}}$ along with the witness $r_i^{\mathbf{L}}$ to the fact that $c_i^{\mathbf{L}}$ is a valid encryption of \mathbf{pw} with respect to pk and l_i . That is, $K_i^{\mathbf{L}} = \mathbf{Hash}(hk_i^{\mathbf{L}}, c_{i-1}^{\mathbf{R}}, l_{i-1}, \mathbf{pw})$ and $K_{i-1}^{\mathbf{R}} = \mathbf{ProjHash}(phk_{i-1}^{\mathbf{R}}, c_i^{\mathbf{L}}, l_i, \mathbf{pw}, r_i^{\mathbf{L}})$. Next, player U_i computes the test master key $X_i^{\mathbf{L}} = K_i^{\mathbf{L}} \cdot K_{i-1}^{\mathbf{R}}$ for its predecessor and verifies if $test_{i-1}^{\mathbf{R}} = \mathbf{UH}_1(X_i^{\mathbf{L}})$. Once again, if this test fails, then player U_i halts and sets $\mathbf{acc}_i = \mathbf{false}$. If this test succeeds, then player U_i computes a test key $test_i^{\mathbf{L}} = \mathbf{UH}_2(X_i^{\mathbf{L}})$ for its predecessor and an auxiliary key $X_i = K_i / K_{i-1}$, where $K_i \triangleq \mathbf{Hash}(hk_i, c_{i+1}^{\mathbf{L}}, l_{i+1}, \mathbf{pw})$. More precisely, player U_i computes the value K_i using the hash key hk_i and the value K_{i-1} using the projection key phk_{i-1} along with the witness $r_i^{\mathbf{L}}$ to the fact that $c_i^{\mathbf{L}}$ is a valid encryption of \mathbf{pw} with respect to pk and l_i . Finally, each player U_i broadcasts the pair $(X_i, test_i^{\mathbf{L}})$.

Round 5. First, each player U_i checks whether $test_{i+1}^{\mathbf{L}} = \mathbf{UH}_2(X_i^{\mathbf{R}})$ and whether $\prod_{l=1}^n X_l = 1$. If any of these tests fails, then player U_i halts and sets $\mathbf{acc}_i = \mathbf{false}$.

Otherwise, each player U_i sets $T_j = vk_j \parallel U_j \parallel c_j \parallel phk_j \parallel phk_j^L \parallel phk_j^R \parallel X_j \parallel X_j^L$ for $j = 1, \dots, n$ and $T = T_1 \parallel \dots \parallel T_n$ and then signs it using sk_i to obtain σ_i . Finally, each player U_i broadcasts σ_i .

Finalization. Each player U_i checks for $j \neq i$ whether σ_j is a valid signature on T with respect to vk_j . If any of these checks fails, then player U_i halts and sets $\text{acc}_i = \text{false}$. Otherwise, player U_i sets $\text{acc}_i = \text{true}$ and computes the master key $MSK = \prod_{j=1}^n K_j = K_i^n \cdot X_{i+1}^{n-1} \cdot X_{i+2}^{n-2} \cdot \dots \cdot X_{i+n-3}^2 \cdot X_{i+n-1}$, and the session key $SK = \text{UH}'(MSK)$. Each player U_i also sets the session identifier sid_i to T .

Observation. Let $K_i \triangleq \text{Hash}(hk_i, c_{i+1}^L, l_{i+1}, \text{pw})$, $K_i^R \triangleq \text{Hash}(hk_i^R, c_{i+1}^L, l_{i+1}, \text{pw})$, and $K_i^L \triangleq \text{Hash}(hk_i^L, c_{i-1}^R, l_{i-1}, \text{pw})$ denote temporary keys. In a normal execution of the protocol, the temporary keys K_i and K_i^R are known to both player U_i (who knows hk_i and hk_i^R) and his successor U_{i+1} (who knows phk_i , phk_i^R , and the witness r_{i+1}^L to the fact that c_{i+1}^L is a valid encryption of pw with respect to pk and l_{i+1}). Likewise, the temporary key K_i^L is known to both player U_i (who knows hk_i^L) and his predecessor U_{i-1} (who knows phk_i^R and the witness r_{i-1}^R to the fact that c_{i-1}^R is a valid encryption of pw with respect to pk and l_{i-1}).

4.2 Correctness and Security

CORRECTNESS. In an honest execution of the protocol, it is easy to verify that all participants in the protocol will terminate by accepting and computing the same values for the partner identifier, session identifiers, and the session key. The session key in this case is equal to $\prod_{j=1}^n \text{Hash}(hk_j, c_{j+1}, l_{j+1}, \text{pw}) = \prod_{j=1}^n K_j$.

SECURITY. The intuition behind the security of our protocol is quite simple. Due to the security properties of the underlying Gennaro-Lindell protocol, each user is able to authenticate its neighbors and safely share session keys with them. Due to the properties of the signature scheme, all users in the group are able to ensure that they had received the same messages and that they will generate the same group session key. As the following theorem shows, the $\mathcal{GP}\mathcal{AKE}$ protocol described above and in Fig. 1 is a secure password-based authenticated group key exchange protocol as long as the primitives on which the protocol is based meet the appropriate security notion described in the theorem.

Theorem 1. *Let \mathcal{LEPKE} be a labeled encryption secure against chosen-ciphertext attacks, let \mathcal{HASH} be a family of smooth projective hash functions, let \mathcal{UH} and \mathcal{UH}' be families of universal hash functions, and let \mathcal{SIG} be a signature scheme that is unforgeable against chosen-message attacks. Let $\mathcal{GP}\mathcal{AKE}$ denote the protocol built from these primitives as described above and let \mathcal{A} be an adversary against $\mathcal{GP}\mathcal{AKE}$. Then, the advantage function $\text{Adv}_{\mathcal{GP}\mathcal{AKE}, \mathcal{A}}^{\text{ake-ind}}(k)$ is only negligibly larger than $O(q/N)$, where q denotes the maximum number of different protocol instances to which \mathcal{A} has asked Send queries and N is the dictionary size.*

The proof can be found in the full version of this paper [2]. In it, we actually show that the security of our protocol is only negligibly larger than $(q_{\text{send-1}} + q_{\text{send-2}})/N$,

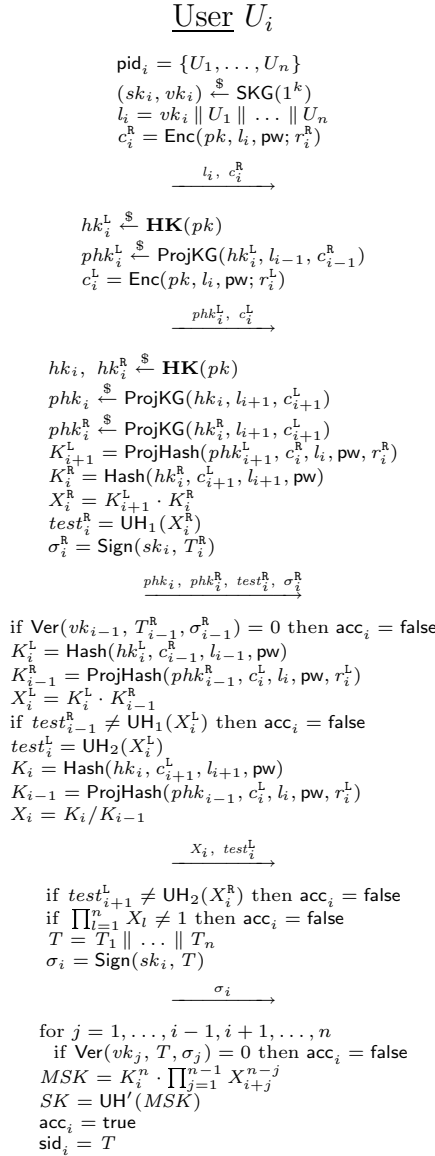


Fig. 1. An honest execution of the password-authenticated group key exchange protocol by player U_i in a group $\{U_1, \dots, U_n\}$, where $T_i^R = U_i \parallel U_{i+1} \parallel c_i^R \parallel c_{i+1}^L \parallel phk_i \parallel phk_i^R \parallel phk_{i+1}^L \parallel test_i^R$ and $T_i = vk_i \parallel U_i \parallel c_i \parallel phk_i \parallel phk_i^L \parallel phk_i^R \parallel X_i \parallel X_i^L$ for $i = 1, \dots, n$

where q_{send-1} and q_{send-2} represent the maximum number of *Send* queries that the adversary can ask with respect to the first and second round of communication and N is dictionary size. Even though we believe this security level is good enough for groups of small to medium sizes, it may not be sufficient in cases where the number of users in a group is large and the dictionary size is small.

In the latter case, it would be desirable to have a scheme whose security is only negligibly larger than the number of sessions (and not protocol instances) over the size of the dictionary. Unfortunately, the latter cannot be achieved by our protocol as it is possible for an active adversary to test in the same session a number of passwords that is linear in the total number of users, for instance by playing the role of every other user.

4.3 Efficiency

Our protocol is quite efficient, only requiring a small amount of computation by each user. In what concerns encryption and hash computations, each user only has to perform 2 encryptions, 3 projection key generations, 3 hash computations, 3 projected hash computations, and 5 universal hash computations. The most expensive part of our protocol, which is linear in the group size, is the number of signature verifications and the master session key computation. While the latter computation can be improved by using algorithms for multi-exponentiations, the former can be improved by using two-time signature schemes.

It is worth mentioning that, as done by Katz et al. [27] in the case of the KOY protocol [28], one could also improve the efficiency of our protocol by using two different encryption schemes when computing the ciphertexts c_i^R and c_i^L broadcasted in the first and second rounds. While the computation of the ciphertexts c_i^R would require a CCA-secure labeled encryption scheme, the computation of the ciphertexts c_i^L would only require a CPA-secure encryption scheme.

4.4 Future Work

One issue not addressed in the current paper is whether our protocol remains secure in the presence of *Corrupt* queries, through which the adversary can learn the values of the long-term secret keys held by a user. This is indeed a significant limitation of our security model which we expect to address in the full version of this paper. In fact, we do hope to be able to prove that our protocol achieves forward security according to the definition given in [30].

Acknowledgements

The authors were supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by France Telecom R&D as part of the contract CIDRE, between France Telecom R&D and École normale supérieure.

References

1. M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-based group key exchange in a constant number of rounds. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 427–442. Springer-Verlag, Berlin, Germany, Apr. 2006.

2. M. Abdalla and D. Pointcheval. A scalable password-based group key exchange protocol in the standard model. Full version of current paper. Available from authors' web pages.
3. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer-Verlag, Berlin, Germany, May 2000.
4. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer-Verlag, Berlin, Germany, May 2000.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
6. M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, Berlin, Germany, Aug. 1994.
7. M. Bellare and P. Rogaway. Provably secure session key distribution — the three party case. In *28th ACM STOC*, pages 57–66. ACM Press, May 1996.
8. S. M. Bellare and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.
9. J.-M. Bohli, M. I. G. Vasco, and R. Steinwandt. Password-authenticated constant-round group key establishment with a common reference string. Cryptology ePrint Archive, Report 2006/214, 2006. <http://eprint.iacr.org/>.
10. V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer-Verlag, Berlin, Germany, May 2000.
11. E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange – the dynamic case. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 290–309. Springer-Verlag, Berlin, Germany, Dec. 2001.
12. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 497–514. Springer-Verlag, Berlin, Germany, Dec. 2002.
13. E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In *ACM CCS 03*, pages 241–250. ACM Press, Oct. 2003.
14. E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. In F. Bao, R. Deng, and J. Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 145–158. Springer-Verlag, Berlin, Germany, Mar. 2004.
15. E. Bresson, O. Chevassut, and D. Pointcheval. A security solution for IEEE 802.11's ad-hoc mode: Password authentication and group Diffie-Hellman key exchange. *International Journal of Wireless and Mobile Computing*, 2005. To appear.
16. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *ACM CCS 01*, pages 255–264. ACM Press, Nov. 2001.
17. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system (extended abstract). In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 275–286. Springer-Verlag, Berlin, Germany, May 1994.
18. M. Burmester and Y. Desmedt. A secure and scalable group key exchange system. *Information Processing Letters*, 94(3):137–143, May 2005.
19. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, Berlin, Germany, Aug. 1998.

20. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer-Verlag, Berlin, Germany, Apr. / May 2002.
21. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
22. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
23. R. Dutta and R. Barua. Password-based encrypted group key agreement. *International Journal of Network Security*, 3(1):30–41, July 2006. <http://isrc.nchu.edu.tw/ijns>
24. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer-Verlag, Berlin, Germany, May 2003. <http://eprint.iacr.org/2003/032.ps.gz>.
25. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.
26. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
27. J. Katz, P. D. MacKenzie, G. Taban, and V. D. Gligor. Two-server password-only authenticated key exchange. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 1–16. Springer-Verlag, Berlin, Germany, June 2005.
28. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer-Verlag, Berlin, Germany, May 2001.
29. J. Katz, R. Ostrovsky, and M. Yung. Forward secrecy in password-only key exchange protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 29–44. Springer-Verlag, Berlin, Germany, Sept. 2002.
30. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125. Springer-Verlag, Berlin, Germany, Aug. 2003.
31. H.-J. Kim, S.-M. Lee, and D. H. Lee. Constant-round authenticated group key exchange for dynamic groups. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 245–259. Springer-Verlag, Berlin, Germany, Dec. 2004.
32. V. Shoup. ISO 18033-2: An emerging standard for public-key encryption. <http://shoup.net/iso/std6.pdf>, Dec. 2004. Final Committee Draft.
33. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, Aug. 2000.

A Weakness in Some Oblivious Transfer and Zero-Knowledge Protocols

Ventzislav Nikov¹, Svetla Nikova², and Bart Preneel²

¹ Philips TASS

`venci.nikov@gmail.com`

² Department Electrical Engineering, ESAT/COSIC,
Katholieke Universiteit Leuven, Kasteelpark Arenberg 10,
B-3001 Heverlee-Leuven, Belgium

`{svetla.nikova, bart.preneel}@esat.kuleuven.be`

Abstract. We consider oblivious transfer protocols and their applications that use underneath semantically secure homomorphic encryption scheme (e.g. Paillier’s). We show that some oblivious transfer protocols and their derivatives such as private matching, oblivious polynomial evaluation and private shared scalar product could be subject to an attack. The same attack can be applied to some non-interactive zero-knowledge arguments which use homomorphic encryption schemes underneath. The roots of our attack lie in the additional property that some semantically secure encryption schemes possess, namely, the decryption also reveals the random coin used for the encryption, and that the (sender’s or prover’s) inputs may belong to a space, that is very small compared to the plaintext space. In this case it appears that even a semi-honest chooser (verifier) can derive from the random coin bounds for all or some of the sender’s (prover’s) private inputs with non-negligible probability. We propose a fix which precludes the attacks.

Keywords: Oblivious Transfer, Homomorphic Semantically Secure Cryptosystems, Paillier’s Public-Key Cryptosystem, Non-Interactive Zero-Knowledge Arguments.

1 Introduction

Oblivious Transfer (OT) [4,30] protocols allow one party, called *the sender* to send part of its inputs to a second party, called *chooser*, in such a manner that the chooser does not receive more information than it is entitled and the sender does not learn which part of the inputs the chooser received. Oblivious transfer is used as a key component in many applications of cryptography.

Naor and Pinkas [26] proposed a way to use OT for *polynomial evaluation*. Another application known as *private matching* solves the problem of two parties who possess lists of items and want to compute their set-intersection or to approximate the size of the intersection. Freedman *et al.* [16] have shown that a simple reduction from oblivious transfer to private matching exists. The authors of [16] used oblivious polynomial evaluation in their solution for the private matching set intersection problem.

In this paper we will work in the *semi-honest security model*, in which the parties follow the protocol, but may be curious. We do not consider malicious parties who may deviate from the protocol. Often, there is no guarantee for the privacy of the sender if the chooser is malicious, but we do not consider this issue.

Our Contribution: We first describe an attack against an oblivious transfer protocol; subsequently we apply the attack to certain protocols derived from oblivious transfer, such as oblivious polynomial evaluation, private matching (set cardinality and subset inclusion) and private (shared) scalar product. For our attack we exploit the additional property that some semantically secure encryption schemes possess, namely that the decryption reveals the random coin used for encryption. We consider the case when the (sender's) inputs belong to a very small space compared to the plaintext space of the Paillier cryptosystem. We show that from the random coin the chooser can derive certain information (bounds) for all (or some) of the sender's private inputs with non-negligible probability. We extend the attack to certain non-interactive zero-knowledge protocols. We introduce the so-called irrational behavior of the chooser, meaning that a semi-honest but curious chooser is "bluffing" in order to get the sender's inputs, i.e. the chooser is putting his privacy at risk. To the best of our knowledge some of the protocols from the following papers [6,12,17,19,31] could be subject to this attack when applied in this scenario. Finally we propose a fix which precludes the attacks.

Organization of the paper: In the next section we introduce the notions of homomorphic semantically secure cryptosystems, oblivious transfer, and different applications of the oblivious transfer. Section 3 provides description of several known protocols and in Sect. 4 the attack against them is proposed. We conclude in Sect. 5.

2 Preliminary

Homomorphic Semantically Secure Cryptosystems

Let $\Pi = (G_\Pi, E, D)$ be a public-key encryption cryptosystem, where G_Π is the key generation algorithm, E is the encryption algorithm and D is the decryption algorithm. Let k be the security parameter, then the key generation algorithm G_Π on input 1^k generates a valid key pair (SK, K) of private and public keys that corresponds to the security parameter k . The encryption algorithm E takes as input a plaintext m , a random coin r and the public key K and outputs the corresponding ciphertext $E_K(m, r)$. The decryption algorithm takes as input a ciphertext c and the private key SK and outputs the corresponding plaintext $D_{SK}(c)$. More formal: $G_\Pi : 1^k \mapsto (SK, K)$; $E_K : (m, r) \mapsto E_K(m, r)$, $D_{SK} : c \mapsto D_{SK}(c)$ and $D_{SK}(c) = m$ if $c = E_K(m, r)$. It is required that $D_{SK}(E_K(m, r)) = m$ for any random coin r , key pair (SK, K) and plaintext m . It is said that Π is *homomorphic*, if $E_K(m_1, r_1) \cdot E_K(m_2, r_2) = E_K(m_1 + m_2, r_1 \cdot r_2)$. It then follows that $E_K(m, r)^s = E_K(s \cdot m, r^s)$.

For an algorithm A , define $Adv_{\Pi,k}^{sem}(A)$ to be the advantage that A has over random guessing when trying to distinguish random encryption of two elements, chosen by herself. It is said that Π is *semantically* secure under an chosen plaintext attack (IND-CPA secure) if for all PPT (probabilistic polynomial time) algorithms A , the advantage $Adv_{\Pi,k}^{sem}(A)$ is negligible in k .

Several homomorphic probabilistic encryption schemes are known: ElGamal [14], Goldwasser and Micali [18], Benaloh [2], Okamoto and Uchiyama [28], Naccache and Stern [25], Paillier [29] and its modifications [5,13].

For the sake of simplicity, we will describe the protocols with the Paillier cryptosystem (some of the protocols which we consider are indeed designed for the Paillier cryptosystem), although most of the homomorphic semantically secure cryptosystems can be used instead of Paillier's. We present the Paillier cryptosystem for completeness, but omit the number-theoretic justifications.

Key Generation: Let N be an RSA modulus $N = pq$, where p, q are large primes. The public key K is N and the secret key SK is $\lambda(N) = lcm((p-1), (q-1))$, where $\lambda(N)$ is the Carmichael function. One can assume w.l.o.g. that $N > 2^k$, where the security parameter $k \geq 1024$.

Encryption: To encrypt a plaintext $m \in \mathbb{Z}_N$, compute the ciphertext

$$c = E_K(m, r) = (1 + mN)r^N \bmod N^2, \text{ with } r \in_R \mathbb{Z}_N^*.$$

Decryption: To decrypt a ciphertext $c \in \mathbb{Z}_{N^2}$, compute the plaintext

$$m = D_{SK}(c) = \frac{L(c^{\lambda(N)} \bmod N^2)}{\lambda(N)} \bmod N, \text{ where } L(u) = \frac{u-1}{N}.$$

The Paillier cryptosystem possesses the following useful properties:

$$\begin{aligned} E_K(m_1, r_1)E_K(m_2, r_2) \bmod N^2 &= E_K(m_1 + m_2 \bmod N, r_1 r_2 \bmod N) \\ E_K(m, r)^s \bmod N^2 &= E_K(sm \bmod N, r^s \bmod N) \\ E_K(m, r)(1 + N)^c \bmod N^2 &= E_K(m + c \bmod N, r). \end{aligned}$$

In order to re-randomize a ciphertext $c = E_K(m, r)$, simply multiply it by a random encryption of 0, i.e. compute $cr_1^N \bmod N^2 = E_K(m, rr_1 \bmod N)$ for $r_1 \in_R \mathbb{Z}_N^*$.

It is well known (see [5]) that for Paillier's cryptosystem $D_{SK}(c) = (m, r)$ if $c = E_K(m, r)$, i.e. the result of the decryption of a ciphertext is the corresponding plaintext and the random coin used for the encryption (usually the random coin cannot be recovered efficiently). Indeed as Catalano *et al.* have shown there is an alternative decryption process based on the observation that the ciphertext $c = E_K(m, r)$ satisfies $c = r^N \bmod N$. The latter can disclose r by an RSA decryption (modulo N , with public exponent N). Now putting r in the original ciphertext equation provides the plaintext m .

We stress here that the ability to efficiently disclose the random coin used for the encryption, forms an essential point for our attack. We pose as an open problem whether our attack can be extended to some of the other homomorphic semantically secure cryptosystems.

2.1 $\binom{n}{1}$ -Oblivious Transfer and Zero-Knowledge Arguments

During an $\binom{n}{1}$ -Oblivious Transfer the *sender* maintains n items and the *chooser* receives one item chosen by him. The sender does not know which item was transferred. The security of an OT is usually defined in two parts. We will follow the definitions of [22,27]. Let \tilde{k} be the security parameter.

Chooser-Privacy: Consider an algorithm A that executes the sender's part of the OT protocol; define $Adv_{Cho,\tilde{k}}^{OT}(A)$ to be the probability that after observing an execution of the protocol, A can predict which choice was made by the chooser. An OT protocol is said to be (computationally) chooser-private if $Adv_{Cho,\tilde{k}}^{OT}(A)$ is negligible for any PPT algorithm A . In all this protocols the chooser-privacy (which holds even against a malicious sender) will be based on the indistinguishability implied by the underlying semantically secure encryption scheme.

Sender-Privacy: Consider an algorithm A executing the chooser's part of the OT protocol; define a simulator S that generates an output that is statistically indistinguishable from the view of A that interacts with the honest sender. More precisely, for an algorithm S define $Adv_{Sen,\tilde{k}}^{OT}(A, S)$ to be the statistical difference of the distributions of the S output and the view of A . An OT protocol is called (statistically) sender-private if for every (not necessarily PPT) A there exists a (not necessarily PPT) S , such that $Adv_{Sen,\tilde{k}}^{OT}(A, S)$ is negligible in \tilde{k} . The sender-privacy is called *perfect* if $Adv_{Sen,\tilde{k}}^{OT}(A, S) = 0$. In all this cases the sender-privacy is based on a comparison with the ideal model.

Recently Damgård *et al.* [12] have proposed a method to build non-interactive zero-knowledge protocols from homomorphic encryption. Namely the authors described a method for compiling a class of Σ -protocols (3-move public-coin protocols) into non-interactive zero-knowledge arguments. In a zero-knowledge proof system a *prover* convinces a *verifier* via an interactive protocol that some statement is true. The verifier should learn nothing beyond the fact that the assumption is valid. Σ -protocols are three-move protocols where conversations are tuples of the form (a, e, z) where e is a random challenge sent by the verifier, a is the prover's input and z is the proof. There are several well-known techniques for making Σ -protocols non-interactive [11,15].

2.2 Applications of OT

As shown by Kilian [21] most cryptographic protocols can be based on oblivious transfer. In this section we will describe several protocols built on top of OT.

An $\binom{n}{1}$ -OT protocol sometimes needs to be *sender-verifiable* (or *committed*) [7,10] in the following sense: the sender commits to every item and sends these commitments to the chooser; these commitments later can be used in various zero-knowledge proofs and arguments.

The notion of *conditional oblivious transfer* (COT) was introduced by Di Crescenzo *et al.* [9]. It is a variant of OT in which the two participants have private inputs, say x and y respectively, and share a public predicate $Q(\cdot, \cdot)$. The sender has a secret s , which is transferred to the chooser if and only if $Q(x, y) = 1$. If $Q(x, y) = 0$, no information about s is transferred to the chooser. The chooser's

private input and the value of the predicate remain computationally hidden from the sender.

The notion of *strong conditional oblivious transfer* (SCOT) has been first introduced by Di Crescenzo [8]; later Blake and Kolesnikov [3] have independently defined the same notion. SCOT strengthens the COT definition, in the SCOT setting – unlike the COT “all-or-nothing” approach – the sender possesses two secrets s_0 and s_1 and transfers s_i if $Q(x, y) = i$ (where $i = 0$ or 1). In addition to the COT requirement that the chooser private input has to be computationally hidden from the sender, the value of the predicate should also remain hidden for both participants.

Consider the following problem: two parties possess lists (sets) of items and they want to compute their set-intersection. Related problems are to approximate the size of the intersection or to decide whether the intersection size is greater than a threshold. Such problems are called *private matching* (PM) in [16]. That is, if the chooser inputs $X = \{x_1, \dots, x_{k_c}\}$ and the sender inputs $Y = \{y_1, \dots, y_{k_s}\}$ then the chooser learns $X \cap Y = \{x_u : \exists v, x_u = y_v\} \leftarrow PM(X, Y)$. The related variants are as follows: the chooser learns $|X \cap Y| \leftarrow PM_C(X, Y)$ for the *intersection size* problem or for the *threshold intersection size* problem he gets $1 \leftarrow PM_t(X, Y)$ if $PM_C(X, Y) > t$ and 0 otherwise. As shown by Freedman *et al.* [16] a simple reduction from oblivious transfer to private matching exists.

In a simpler form of PM both lists contain just one item, thus the two parties want to compare their private inputs without leaking it. *Private equality test* (PET) allows the chooser to know whether his private input and the sender’s private input are equal [16,22].

Another kind of PM is the *private subset inclusion*. Namely, both participants have sets X and Y as inputs and the chooser gets 0 if $X \subseteq Y$ or 1 otherwise. Laur *et al.* [24] have proposed a private subset inclusion protocol, based on an improvement of the intersection size protocol by Freedman *et al.* [16].

Naor and Pinkas [26] proposed a way to use OT for *polynomial evaluation* (OPE). Freedman *et al.* [16] used OPE in their solution for the PM set intersection problem. Recently Freedman *et al.* [17] proposed another OPE protocol which is used as a building block for a *keyword search* protocol.

A protocol between two parties is called a *scalar product* (SP) protocol when on private inputs of both parties $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$ it outputs their scalar product $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$. A protocol is called a *shared scalar product* (SSP) protocol [19] when both parties receive as output of the protocol uniformly distributed additive shares of the scalar product, i.e., the chooser gets $s_c \in \mathbb{Z}_N$ and the sender gets $s_s \in \mathbb{Z}_N$ such that $s_c + s_s = \langle \mathbf{x}, \mathbf{y} \rangle \pmod N$. These protocols are called *private* if the inputs (i.e. \mathbf{x} and \mathbf{y}) are not disclosed.

3 The Protocols

This section describes the protocols to which our attacks can be applied. The reader who is familiar with these protocols can skip this section and continue with the attack described in Sect. 4.

Consider the standard OT setting, i.e. the chooser and the sender have their private inputs. The chooser encrypts his input and sends it to the sender. The sender applies a transformation to the encrypted chooser's input and to his own input (which could be also encrypted). The value obtained in this way is returned to the chooser.

3.1 $\binom{n}{1}$ -Oblivious Transfer

We will start with a short description of Homomorphic Oblivious Transfer and the AIR protocol [1,22].

Private Inputs:

- The sender has a vector $\mu = (\mu_1, \dots, \mu_n)$, $\mu_i \in \mathbb{Z}_T$ and $T \leq N$.
- The chooser has made a choice $\sigma \in \{1, \dots, n\}$.

Private Output: The chooser gets μ_σ .

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. Then generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c \leftarrow E_K(\sigma, r)$. He sends K and c to the sender.
2. For $i = 1, \dots, n$ the sender performs the following: generates random coins $r_i, s_i \in_R \mathbb{Z}_N^*$ and computes $c_i \leftarrow E_K(\mu_i, 1) (c E_K(-i, 1))^{s_i} E_K(0, r_i) \bmod N^2$. He sends c_1, \dots, c_n to the chooser.
3. The chooser obtains $\mu_\sigma \leftarrow D_{SK}(c_\sigma)$.

Homomorphic $\binom{n}{1}$ -Oblivious Transfer

Aiello *et al.* [1] have proposed an OT protocol, which provides perfect sender-privacy and computational chooser-privacy (AIR protocol, in short). This protocol has been slightly modified and generalized by Lipmaa [22] to a *homomorphic oblivious transfer* (HOT) protocol. In [23] the authors fix some problems with the scheme from [22].

Since the encryption scheme is semantically secure, the sender cannot derive σ from the ciphertext c (step 1), which guarantees the chooser-privacy. Using the homomorphic property of the encryption scheme it is easy to verify that in step 2 the sender computes $c_i \leftarrow E_K(\mu_i + (\sigma - i)s_i \bmod N, r_i r^{s_i} \bmod N)$. Then in step 3 the chooser can obtain $\mu_i + (\sigma - i)s_i \bmod N$. But since the s_i are random coins, the values μ_i are perfectly hidden, except μ_σ . This guarantees the correctness of the scheme and the sender-privacy. The HOT protocol is further used in [22] to build committed OT and PET protocols.

Stern's $\binom{n}{1}$ -Oblivious Transfer

Now we present the OT protocol proposed by Stern [31]; this protocol has later been rediscovered by Chang [6]. The original protocol uses a homomorphic semantically secure encryption scheme and a homomorphic commitment scheme. The Paillier encryption scheme, proposed one year after the publication of [31], is not used in the original scheme.

Private Inputs:

- The sender has a vector $\mu = (\mu_1, \dots, \mu_n)$, $\mu_i \in \mathbb{Z}_T$ and $T \leq N$.
- The chooser has made a choice $\sigma \in \{1, \dots, n\}$.

Private Output: The chooser gets μ_σ .

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. He chooses an n -tuple (x_1, \dots, x_n) such that $x_\sigma = 1$ and $x_i = 0$ for $i \neq \sigma$. Then generates n random coins $r_i \in_R \mathbb{Z}_N^*$ and computes $c_i \leftarrow E_K(x_i, r_i)$ for $i = 1, \dots, n$. He sends K and c_1, \dots, c_n to the sender. Last he provides zero-knowledge proofs that all x_i except one are equal to 0 and the nonzero one is equal to 1.
2. The sender generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c \leftarrow (\prod_{i=1}^n c_i^{\mu_i}) E_K(0, r) \bmod N^2$. He sends c to the chooser.
3. The chooser obtains $\mu_\sigma \leftarrow D_{SK}(c)$.

Using the homomorphic property of the encryption scheme it is easy to verify that in step 2 the sender computes $c \leftarrow E_K(\sum_{i=1}^n \mu_i x_i \bmod N, r \prod_{i=1}^n r_i^{\mu_i} \bmod N)$. Then in step 3 the chooser can obtain $\bar{\mu} = \sum_{i=1}^n \mu_i x_i \bmod N$. But since (x_1, \dots, x_n) is such that $x_\sigma = 1$ and $x_i = 0$ for $i \neq \sigma$ the decrypted value is $\bar{\mu} = \mu_\sigma$.

Note that in both OT protocols [31] and [1,22] the sender uses an encryption of 0 (step 2) to re-randomize the ciphertext.

3.2 Oblivious Polynomial Evaluation

Recall that oblivious polynomial evaluation protocol is a building block for other more complex protocols, for example private matching. The protocol given by Freedman *et al.* [17] can be described as follows.

Private Inputs:

- The chooser input is a value $\bar{x} \in \mathbb{Z}_T$.
- The sender input is a polynomial $P(x) = \sum_{i=0}^n a_i x^i$, $a_i \in \mathbb{Z}_T$.
- T is chosen such that $\max(|P(x)|) \leq N$.

Private Output: The chooser gets $P(\bar{x})$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. Then he generates random coins $r_j \in_R \mathbb{Z}_N^*$ and computes $c_j \leftarrow E_K(\bar{x}^j, r_j)$ for $j = 1, \dots, n$. The chooser sends K and c_1, \dots, c_n to the sender.
2. The sender generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c = E_K(a_0, r)(\prod_{j=1}^n c_j^{a_j}) \bmod N^2$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts, i.e. he computes $z = D_{SK}(c)$.

Observe that $c = E_K(\sum_{j=0}^n a_j \bar{x}^j \bmod N, r \prod_{j=1}^n r_j^{a_j} \bmod N)$, thus $z = \sum_{j=0}^n a_j \bar{x}^j \bmod N$, i.e. $z = P(\bar{x})$. Note that the sender re-randomizes the ciphertext (step 2) in a slightly non standard way – by encrypting a_0 with a random r instead of encrypting 0 afterwards.

3.3 Private Shared Scalar Product

In [19] Goethals *et al.* proposed a private SSP protocol. As pointed out by the authors of [19] a private SP can be obtained immediately from the private SSP protocol by defining $s_s \leftarrow 0$. We present here the private SSP protocol.

Private Inputs:

- The chooser input is a vector $\mathbf{x} = (x_1, \dots, x_n)$, $x_i \in \mathbb{Z}_T$ and $T \leq \lfloor \sqrt{N/n} \rfloor$.
- The sender input is a vector $\mathbf{y} = (y_1, \dots, y_n)$, $y_i \in \mathbb{Z}_T$.

Private Output:

- The chooser gets a share $s_c \in \mathbb{Z}_N$.
- The sender gets a share $s_s \in \mathbb{Z}_N$.
- Such that $s_c + s_s = \langle \mathbf{x}, \mathbf{y} \rangle \pmod N$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_{\Pi}(1^k)$. He generates a random coin $r_i \in_R \mathbb{Z}_N^*$ and computes $c_i \leftarrow E_K(x_i, r_i)$ for $i = 1, \dots, n$. The chooser sends K and c_1, \dots, c_n to the sender.
2. The sender performs the following: generates a random coin $r \in_R \mathbb{Z}_N^*$, a random share $s_s \in_R \mathbb{Z}_N$ and computes $c \leftarrow E_K(-s_s, 1)(\prod_{i=1}^n c_i^{y_i})E_K(0, r) \pmod{N^2}$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts and sets it as his share s_c , i.e. he computes $s_c = D_{SK}(c)$.

Note that $c = E_K(-s_s + \sum_{i=1}^n x_i y_i \pmod N, r \prod_{i=1}^n r_i^{y_i} \pmod N)$, thus $s_c = -s_s + \langle \mathbf{x}, \mathbf{y} \rangle \pmod N$, i.e. the protocol is correct. Again the semantic security of the encryption scheme guarantees the chooser-privacy. The sender-privacy is preserved since the chooser only sees a random encryption of $-s_s + \langle \mathbf{x}, \mathbf{y} \rangle$, where s_s is random. Note again that the sender uses encryption of 0 (step 2) to re-randomize the ciphertext.

The authors of [19] give an interesting application of an SP protocol: if $x_i, y_i \in \{0, 1\}$, i.e. \mathbf{x} and \mathbf{y} are the characteristic vectors of two sets X and Y , then $\langle \mathbf{x}, \mathbf{y} \rangle = |X \cap Y|$. In other words such an SP protocol provides solution for the private matching intersection set size problem.

3.4 Private Matching

We first describe the private subset inclusion protocol given by Laur *et al.* [24]. Then we propose a modification to this protocol, which is more efficient.

Laur's Private Subset Inclusion

The authors of [24] use the fact that $X \subseteq Y$ if and only if $|X| = |X \cap Y|$. Instead of using directly the sets, their characteristic functions (denoted with the same letters) are used in the protocol, where $X[i] = 1$ if $i \in X$ and $X[i] = 0$ otherwise ($Y[i]$ is defined in a similar way).

Private Inputs:

- The chooser input is a set $X \subseteq \{1, \dots, n\}$.
- The sender input is a set $Y \subseteq \{1, \dots, n\}$.

Private Output: The chooser gets 0 if $X \subseteq Y$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_{\Pi}(1^k)$. Then he generates a random coin $r_j \in_R \mathbb{Z}_N^*$ and computes $c_j \leftarrow E_K(X[j], r_j)$ for $j = 1, \dots, n$. The chooser sends K and c_1, \dots, c_n to the sender.
2. The sender generates random coins $r, s \in_R \mathbb{Z}_N^*$ and computes $c = (\prod_{j:Y[j]=0} c_j)^s E_K(0, r) \bmod N^2$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts, i.e. he computes $z = D_{SK}(c)$ and accepts that $X \subseteq Y$ if $z = 0$.

Note that $c = E_K(s \sum_{j:Y[j]=0} X[j] \bmod N, r \prod_{j:Y[j]=0} r_j^s \bmod N)$. Thus the chooser gets $z = s \sum_{j:Y[j]=0} X[j] \bmod N$, which is zero only if all $X[j] = 0$ when $Y[j] = 0$. The last relation implies that $X \subseteq Y$.

Private Subset Inclusion

We also do not use directly the sets in our protocol, but their characteristic functions redefined as follows $X[i] = s_i$ if $i \in X$ (for a random nonzero $s_i \in_R \mathbb{Z}_T^*$ and $T \leq \lfloor N/n \rfloor$) and $X[i] = 0$ otherwise.

Private Inputs:

- The chooser input is a set $X \subseteq \{1, \dots, n\}$.
- The sender input is a set $Y \subseteq \{1, \dots, n\}$.

Private Output: The chooser gets 0 if $X \subseteq Y$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_{\Pi}(1^k)$. Then he generates a random coin $r_j \in_R \mathbb{Z}_N^*$ and computes $c_j \leftarrow E_K(X[j], r_j)$ for $j = 1, \dots, n$. The chooser sends K and c_1, \dots, c_n to the sender.
2. The sender generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c = (\prod_{j:Y[j]=0} c_j) E_K(0, r) \bmod N^2$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts, i.e. he computes $z = D_{SK}(c)$ and accepts that $X \subseteq Y$ if $z = 0$.

Note that $c = E_K(\sum_{j:Y[j]=0} X[j] \bmod N, r \prod_{j:Y[j]=0} r_j \bmod N)$. Thus $z = \sum_{j:Y[j]=0} X[j]$, which is zero only if all $X[j] = 0$ when $Y[j] = 0$. The last relation implies that $X \subseteq Y$. Obviously this protocol is more efficient than the original protocol of [24] since the sender does not need to compute a random power of $\prod_{j:Y[j]=0} c_j$. Note again that the standard way to re-randomize the ciphertext (step 2) is used in both protocols, i.e. the sender uses an encryption of 0.

3.5 Zero-Knowledge Arguments

Consider the following protocol for equality of double base discrete logarithms. We consider another Σ -protocol than the one in [12] which is for the equality of discrete logarithms, where the prover should prove that indeed $h_1 = g_1^w \bmod p$ and $h_2 = g_2^w \bmod p$ for some w . Let \tilde{k} be the security parameter.

Input:

- The system setting is the tuple $(p, p', g_1, g_2, h_1, h_2)$ where p, p' are prime, p' is \tilde{k} -bit long, $p = 2p' + 1$, $g_1 \in \mathbb{Z}_p^*$ has order p' and $g_2, h_1, h_2 \in \langle g_1 \rangle$. In addition $g_2 = g_1^y$ for some secret $y \in \mathbb{Z}_p^*$ and $h_1 = g_1^w g_2^{w_1}, h_2 = g_1^w g_2^{w_2}$ for some $w, w_1, w_2 \in \mathbb{Z}_p^*$.
- The tuple $(p, p', g_1, g_2, h_1, h_2)$ is a common input to the prover and the verifier.
- The prover gets w, w_1, w_2 as private input.

Output: The verifier checks whether $\log_{g_1}(h_1) \bmod y = \log_{g_2}(h_2) \bmod y$.

1. The prover chooses random $3\tilde{k}$ -bit integers r, r_1, r_2 and sends $a_1 = g_1^r g_2^{r_1} \bmod p$ and $a_2 = g_1^r g_2^{r_2} \bmod p$ to the verifier.
2. The verifier chooses the challenge e at random in $\mathbb{Z}_{p'}$ and sends it to the prover.
3. The prover computes $z = r + ew, z_1 = r_1 + ew_1, z_2 = r_2 + ew_2$ and sends them to the verifier who checks that $g_1^z g_2^{z_1} = a_1 h_1^e \bmod p$ and $g_1^z g_2^{z_2} = a_2 h_2^e \bmod p$.

4 The Proposed Attack

4.1 Attack Against Oblivious Transfer

We first specify the information that the chooser possesses after finishing the protocol.

- Consider the Stern’s OT protocol described in Section 3.1. Denote by $\bar{r} = r \prod_{i=1}^n r_i^{\mu_i} \bmod N$ and recall that $D_{SK}(c) = (\bar{\mu}, \bar{r})$, where $\bar{\mu} = \sum_{i=1}^n \mu_i x_i \bmod N$. Thus the chooser obtains $\bar{\mu}$ and \bar{r} .
- Consider the OPE protocol described in Section 3.2. Denote by $\bar{r} = r \prod_{j=1}^n r_j^{a_j} \bmod N$ and recall that $D_{SK}(c) = (z, \bar{r})$, where $z = P(\bar{x})$. Thus the chooser obtains z and \bar{r} .
- Consider the private SSP protocol described in Section 3.3. Recall that $D_{SK}(c) = (\bar{m}, \bar{r})$, where $\bar{m} = -s_s + \sum_{i=1}^n x_i y_i \bmod N$ and $\bar{r} = r \prod_{i=1}^n r_i^{y_i} \bmod N$. Thus the chooser obtains \bar{m} and \bar{r} .
- Consider the modified Subset Inclusion protocol described in Section 3.4. Recall that $D_{SK}(c) = (z, \bar{r})$, where $z = \prod_{j:Y[j]=0} X[j] \bmod N$ and $\bar{r} = r \prod_{j:Y[j]=0} r_j \bmod N$. Thus the chooser obtains \bar{z} and \bar{r} .

Notice that in all these cases \bar{r} has a common form, which we will further unify as $\bar{r} = r \prod_{i=1}^n r_i^{y_i} \bmod N$.

Scenario

Now we describe the scenario in which our attack can be mounted by the chooser. Recall that the sender’s inputs to the protocol are $y_i \in \mathbb{Z}_T$. We consider the case when $T \ll N$, i.e. is very small; how small will be specified later. In this case a semi-honest chooser with *irrational* behavior can try to get some information on the sender’s inputs with a non-negligible probability. For the sake of simplicity

we only consider the case of a uniform probability distribution for (y_1, \dots, y_n) , but our results hold for any probability distribution.

Attack - Phase 1

Let the chooser select r_i in step 1 to be small prime numbers, e.g. $2 \leq p_1 \leq p_2 \leq \dots \leq p_n \ll N$. Thus the probability that $\gcd(r_i, r) \neq 1$ for some i is $\frac{1}{r_i}$, when $r \in_R \mathbb{Z}_N^*$ is chosen (independently) by the sender in step 2. Hence the probability $P_i = Pr[r_i = p_i \text{ and } r \in_R \mathbb{Z}_N^* : \gcd(r, r_i) \neq 1] = 1/p_i$.

Consider the random coin \bar{r} obtained by the chooser after decrypting the sender's reply. Denote by $\tilde{r} = r \prod_{i=1}^n r_i^{y_i}$ thus $\tilde{r} = \bar{r} + \ell N$, where $\ell = 0, 1, \dots$. Recall that $y_i \in \mathbb{Z}_T$, $r \in_R \mathbb{Z}_N^*$ and $r_i = p_i$. Denote by $\bar{N} = (\prod_{i=1}^n p_i)^{T-1}$ hence $\ell < \bar{N}$. Denote by $x = \frac{N}{\prod_{i=1}^n p_i^{y_i}}$ (assuming the y_i 's are fixed) then $Pr[r \in_R \mathbb{Z}_N^* : r < x] = \frac{x}{N}$ and since the probability that (y_1, \dots, y_n) is the concrete sender's input is $\frac{1}{T^n}$ we obtain that

$$\begin{aligned}
 P[\ell = 0] &= Pr[y_i \in_R \mathbb{Z}_T, r \in_R \mathbb{Z}_N^* : r \prod_{i=1}^n p_i^{y_i} < N] \tag{1} \\
 &= \sum_{(\bar{y}_1, \dots, \bar{y}_n)} Pr[(y_1, \dots, y_n) = (\bar{y}_1, \dots, \bar{y}_n)] Pr[r \in_R \mathbb{Z}_N^* : r \prod_{i=1}^n p_i^{\bar{y}_i} < N] \\
 &= \frac{1}{T^n} \sum_{(\bar{y}_1, \dots, \bar{y}_n)} \frac{1}{\prod_{i=1}^n p_i^{\bar{y}_i}} = \frac{1}{T^n} \frac{\prod_{i=1}^n (p_i^T - 1)}{\prod_{i=1}^n p_i^{T-1} (p_i - 1)} > \frac{1}{T^n}.
 \end{aligned}$$

Notice that $2x < N$ when $(y_1, \dots, y_n) \neq (0, \dots, 0)$ and $x = N$ when $(y_1, \dots, y_n) = (0, \dots, 0)$, thus we obtain $Pr[r \in_R \mathbb{Z}_N^*, x \neq N : x \leq r < 2x] = \frac{x}{N}$. It can be observed that $P[\ell = 0] > P[\ell = i]$ for any $i > 0$, for example:

$$\begin{aligned}
 P[\ell = 1] &= Pr[y_i \in_R \mathbb{Z}_T, r \in_R \mathbb{Z}_N^* : N \leq r \prod_{i=1}^n p_i^{y_i} < 2N] \\
 &= \frac{1}{T^n} \sum_{(\bar{y}_1, \dots, \bar{y}_n) \neq (0, \dots, 0)} \frac{1}{\prod_{i=1}^n p_i^{\bar{y}_i}} = \frac{1}{T^n} \left(\frac{\prod_{i=1}^n (p_i^T - 1)}{\prod_{i=1}^n p_i^{T-1} (p_i - 1)} - 1 \right).
 \end{aligned}$$

Hence $P[\ell = 1] = P[\ell = 0] - \frac{1}{T^n}$. More importantly $P[\ell = 0]$ depends only on the primes selected by the chooser and the system parameters n and T .

Attack - Phase 2

Now we explain further how the attack works. The protocol is executed just once with the exception that the chooser does not generate the r_i at random but instead selects them as described above. At the end of the execution the chooser possesses \bar{r} and with probability $P[\ell = 0]$ he guesses \tilde{r} . Note that $r_i^{y_i}$ is a factor of \tilde{r} co-prime with the other factors, except maybe with r . Let the attacker target some of the secrets y_i for $i \in I$ ($I \subseteq [n] = \{1, \dots, n\}$). We stress that the chooser should select different prime numbers p_i for $i \in I$. Thus from \tilde{r} the chooser can find $\bar{y}_i, i \in I$, by simple division. Hence $y_i \leq \bar{y}_i$ holds, moreover the difference between \bar{y}_i and y_i is equal to the power m_i of p_i , such that $p_i^{m_i}$

divides r but $p_i^{m_i+1}$ doesn't. Thus an irrational semi-honest chooser can derive from \tilde{r} upper bounds for all y_i for $i \in I$ with probability $P[\ell = 0]$.

Stern's OT protocol and the modified Subset Inclusion protocol give the chooser additional information namely μ_σ (z respectively) which can be used to verify the derived values $\overline{y_i}$. If there is a mismatch between them (e.g. $\mu_\sigma > \overline{y_\sigma}$) then the chooser tries the next \tilde{r} for $\ell = 1$ (with probability $P[\ell = 1]$) and so on.

Setting

We are in position now to clarify the setting of our attack (i.e. when it is feasible at all) and more precisely what we mean by T to be small (i.e. $T \ll N$). We recall here that the security parameter \tilde{k} for the OT is the logarithm of $\frac{1}{T^n}$. The other security parameter k ensures only that the Paillier cryptosystem is secure and in this case $\frac{1}{2^k} = \frac{1}{N} \ll \frac{1}{T^n}$ (i.e. $k \geq \tilde{k}$) holds, i.e. we consider the case when $\frac{1}{T^n}$ is non-negligible in k . Note that in some protocols it is implicitly assumed that $T = N$, but sometimes this requirement is not imposed. We want to point out that for all four protocols T is allowed to be small, moreover for the private SSP (used for PM intersection set problem) and the modified Subset Inclusion protocols we have explicitly $T = 2$.

Recall that the chooser derives with probability $P[\ell = 0]$ upper bounds for all y_i for $i \in I$, i.e. $y_i \leq \overline{y_i}$. Hence to break the security of the protocol, namely the sender's privacy, it is sufficient that $P[\ell = 0] > \frac{1}{T^n}$ (i.e. for $I = [n]$). Indeed the inequality holds, see (1). Thus the attacker obtains upper bounds for the secrets, which contradicts the security goal of the protocol.

Now we will show that if the attacker tries to find the exact values y_i for some set I his success probability is negligible. The attack success probability P of finding the exact values y_i is the product of the probability $P[\ell = 0]$ and the probabilities of $gcd(r_i, r) = 1$ for those $y_i, i \in I$, i.e.

$$\begin{aligned}
 P &= P[\ell = 0] \prod_{i \in I} (1 - P_i) = \frac{1}{T^n} \prod_{i=1}^n \frac{p_i^T - 1}{p_i^{T-1}(p_i - 1)} \prod_{i \in I} \frac{p_i - 1}{p_i} \\
 &= \frac{1}{T^n} \prod_{i \in I} \frac{p_i^T - 1}{p_i^T} \prod_{i \in [n] \setminus I} \frac{(p_i^T - 1)}{p_i^{T-1}(p_i - 1)} = \frac{1}{T^n} \prod_{i=1}^n \left(1 - \frac{1}{p_i^T}\right) \prod_{i \in [n] \setminus I} \frac{p_i}{p_i - 1}.
 \end{aligned}$$

Obviously the higher P is, the more successful the attack. In order to get the exact values of $y_i, i \in I$, it is sufficient that $P > \frac{1}{T^{|I|}}$ (the random guessing), but it is easy to verify that

$$P = \frac{1}{T^n} \prod_{i=1}^n \left(1 - \frac{1}{p_i^T}\right) \prod_{i \in [n] \setminus I} \frac{p_i}{p_i - 1} \leq \frac{1}{T^n} \prod_{i=1}^n \left(1 - \frac{1}{p_i^T}\right) 2^{n-|I|} < \frac{1}{T^{|I|}}$$

because $T \geq 2$ and taking $p_i = 2$ for $i \in [n] \setminus I$. Hence the success probability of this attack is indeed negligible.

But the attacker still can mount a stronger attack than finding upper bounds for the secrets. Since the probability $\overline{P_i} = Pr[r_i = p_i \text{ and } r \in_R \mathbb{Z}_N^* : gcd(r, r_i) \neq 1 \text{ and } gcd(r/r_i, r_i) \neq 1] = \frac{1}{p_i^2}$ the attacker obtains with probability

$\bar{P} = P[\ell = 0] \prod_{i \in I} (1 - \bar{P}_i)$ that $y_i \in \{\bar{y}_i - 1, \bar{y}_i\}$ for $i \in I$. This is the probability that $m_i \in \{0, 1\}$. Indeed when $I = [n]$ it is easy to check that

$$\bar{P} = \frac{1}{T^n} \frac{\prod_{i=1}^n (p_i^T - 1)}{\prod_{i=1}^n p_i^{T-1} (p_i - 1)} \prod_{i=1}^n \frac{(p_i^2 - 1)}{p_i^2} > \frac{1}{T^n}$$

holds since $p_i \geq 2$ and $T \geq 2$. Thus with probability \bar{P} better than random guessing the attacker derives sets with two values for each of the secrets, which is particularly interesting when $T > 2$.

To summarize, we have proved the inequalities: $P[\ell = 0] > \bar{P} > \frac{1}{T^n} > P$; and we have shown that $y_i \leq \bar{y}_i$ with probability $P[\ell = 0]$ and $y_i \geq \bar{y}_i - 1$ with probability \bar{P} .

Example

Let the system parameters are $T = 5$ and $n = 2$. If the attacker selects $p_1 = 2$ and $p_2 = 3$ the success probabilities of the attacks are as follows: $P[\ell = 0] = \frac{2.89429}{25}$, $P[\ell = 1] = \frac{1.89429}{25}$, $\bar{P} = \frac{1.92953}{25}$ and $P = \frac{0.96476}{25}$ while the random guessing has probability $\frac{1}{25}$. Thus with approximately three times better probability than random guessing the attacker obtains the upper bound and with approximately twice better probability the lower bound for each secret.

Discussion

A natural question is why this attack doesn't apply to the HOT and AIR protocols? Recall that $D_{SK}(c_i) = (\bar{\mu}_i, \bar{r}_i)$, where $\bar{\mu}_i = \mu_i + (\sigma - i)s_i \text{ mod } N$ and $\bar{r}_i = r_i r^{s_i} \text{ mod } N$. But now since the sender chooses r and s_i at random in \mathbb{Z}_N the chooser can not derive s_i from \bar{r}_i . The same trick precludes the attack in the original Subset Inclusion protocol described in Section 3.4.

Now we clarify why we call the chooser *irrational*. Note that in order to mount the attack the chooser puts his privacy at risk. This happens because the Paillier cryptosystem is not secure if the used "random coin" is not random. It can be easily verified that if the attacker knows r then he can efficiently reveal m from $E_k(m, r)$. Thus if the sender knows that he is subject to an attack he can reveal the chooser's private input(s). Thus in order to get the sender's inputs the chooser has to bluff, which we call *irrational* behavior.

Our attack does not contradict the semantic security of the Paillier cryptosystem since the attack is performed by the owner of the private key. More precisely the owner of the private key encrypts a message which is then modified by another entity and returned back to the owner, who decrypts it and tries to figure out what the modification was. We would like to point out that the additional information from the random coins affects OT protocols because of their specific nature.

4.2 Attack Against Non-interactive Zero-Knowledge

We apply the compilation technique from [12] to obtain from the zero-knowledge protocol described in Section 3.5 a non-interactive one. Then we show that in this different (compare to OT protocols) scenario our attack can be mounted too.

Setting

Input:

- The system setting is the tuple $(p, p', g_1, g_2, h_1, h_2)$ where p, p' are prime, p' is \tilde{k} -bit long, $p = 2p' + 1$, $g_1 \in \mathbb{Z}_p^*$ has order p' and $g_2, h_1, h_2 \in \langle g_1 \rangle$. In addition $g_2 = g_1^y$ for some secret $y \in \mathbb{Z}_p^*$ and $h_1 = g_1^w g_2^{w_1}, h_2 = g_1^w g_2^{w_2}$ for some $w_1, w_2 \in \mathbb{Z}_p^*$. Let $w \in \mathbb{Z}_T$ and $T \ll N$.
- The tuple $(p, p', g_1, g_2, h_1, h_2)$ is a common input to the prover and the verifier.
- The prover gets w, w_1, w_2 as private input.
- The verifier generates a (private, public) key-pair $(SK, K) \leftarrow G_{\Pi}(1^k)$. Then he generates a random challenge $e \in_R \mathbb{Z}_N^*$, a random coin $s \in_R \mathbb{Z}_N^*$ and computes $\tilde{c} \leftarrow E_K(e, s)$.
- The prover gets \tilde{c} and K as input.

Output: The verifier checks whether $\log_{g_1}(h_1) \bmod y = \log_{g_2}(h_2) \bmod y$.

Protocol Compile

1. The prover chooses random $3\tilde{k}$ -bit integers r, r_1, r_2 and computes $a_1 = g_1^r g_2^{r_1} \bmod p$ and $a_2 = g_1^r g_2^{r_2} \bmod p$.
2. The prover computes $c = E_K(r, \tilde{s})\tilde{c}^w, c_1 = E_K(r_1, \tilde{s}_1)\tilde{c}^{w_1}, c_2 = E_K(r_2, \tilde{s}_2)\tilde{c}^{w_2}$ with some random coins $\tilde{s}, \tilde{s}_1, \tilde{s}_2 \in_R \mathbb{Z}_N^*$. His proof is the tuple (a_1, a_2, c, c_1, c_2) .

Verification

1. The verifier decrypts c, c_1, c_2 obtaining $D_{SK}(c) = (z, \bar{r}), D_{SK}(c_1) = (z_1, \bar{r}_1), D_{SK}(c_2) = (z_2, \bar{r}_2)$. Where $z = r + ew, z_1 = r_1 + ew_1, z_2 = r_2 + ew_2$.
2. Then the verifier checks that $g_1^z g_2^{z_1} = a_1 h_1^e \bmod p$ and $g_1^z g_2^{z_2} = a_2 h_2^e \bmod p$.

Note that the ciphertexts c, c_1, c_2 are randomized by the prover. The verifier can mount the attack as follows. Let us emphasize that we explicitly require $w \in \mathbb{Z}_T$ and $T \ll N$. It is easy to compute that $\bar{r} = \tilde{s}s^w, \bar{r}_1 = \tilde{s}_1 s^{w_1}, \bar{r}_2 = \tilde{s}_2 s^{w_2}$. In the setting phase the verifier chooses s to be a small prime number e.g. p_1 . Thus the probability that $\gcd(s, \tilde{s}) \neq 1$ is $\frac{1}{p_1}$. Moreover since $\tilde{s} \in_R \mathbb{Z}_N^*$ the probability $Pr[\tilde{s}, w : \tilde{s}s^w < N] = \frac{1}{T} \frac{(p_1^T - 1)}{p_1^{T-1}(p_1 - 1)}$ is larger than $\frac{1}{T}$. Hence the same type of attack can again be mounted by the verifier in the verification phase if the space from which w is selected is small. Bound for w can be derived, but not the exact value. Note that we intentionally modified the protocol from [12] to the Pedersen commitment, since the Pedersen commitment can perfectly hide any (even a small) secret w (by w_1 and w_2).

4.3 Precluding the Attack

Finally we propose an easy fix to the protocols in order to resist our attack. Note that all these protocols use an encryption of 0 to re-randomize the ciphertext. If more than one re-randomization is applied (e.g. two) then the probability $\bar{P}[\ell = 0] = Pr[r, s \in_R \mathbb{Z}_N^* : rs \prod_{i=1}^n p_i^{y_i} < N]$ is smaller than the probability $P[\ell = 0]$ (OT case) multiplied by $\frac{\sum_{i=1}^N 1/i}{N} \approx \frac{\ln(N)}{N}$ and therefore becomes negligible. The

probability can be computed by taking into account that $Pr[r, s \in_R \mathbb{Z}_N^* : rs < x] = \sum_{i=1}^{x-1} Pr[s = i] Pr[r < x/i] = \frac{1}{N} \sum_{i=1}^{x-1} \frac{x}{iN} = \frac{x}{N^2} \sum_{i=1}^{x-1} \frac{1}{i} < \frac{x}{N^2} \sum_{i=1}^N \frac{1}{i}$. Thus we have shown that in these settings just one re-randomization is not sufficient, but two (or more) suffice.

5 Conclusion

We have described an attack against several OT protocols and protocols derived from OT such as private matching, oblivious polynomial evaluation and private shared scalar product, which are based on semantically secure homomorphic encryption scheme (e.g. Paillier's). Some semantically secure encryption schemes possess the additional property (e.g. Paillier's) – that they also decrypt the random coin used for the encryption. We have shown that in certain cases the information which can be derived from the random coin is sufficient even for a semi-honest chooser to obtain bounds for the sender's private inputs with non-negligible probability.

The following protocols could be subject to this attack: Stern at Asiacypt'98, Goethals *et al.* at ICISC'04, Chang at ACISP'04, Freedman *et al.* at TCC'05, Damgard *et al.* at TCC'06 if applied in the scenario, when the secrets belong to a space very small compared to the (Paillier's) plaintext space. A fix which precludes the attacks is proposed.

Acknowledgements. We would like to thank the anonymous reviewers of AC 2006 for the very helpful comments and suggestions.

References

1. B. Aiello, Y. Ishai, O. Reingold. Priced Oblivious Transfer: How to Sell Digital Goods, *EUROCRYPT'01*, LNCS 2045, 2001, B. Pfitzmann (Ed.), pp. 119–135.
2. J. Benaloh. Verifiable Secret-Ballot Elections, *Ph.D. Thesis*, Yale's Univ., 1987.
3. I. Blake, V. Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals, *ASIACRYPT'04*, LNCS 3329, 2004, Lee, Pil Joong (Ed.), pp. 515–529.
4. G. Brassard, C. Crepeau, J. Robert. All-or-nothing disclosure of secrets, *Crypto'86*, LNCS 263, 1987, pp. 234–238.
5. D. Catalano, R. Gennaro, N. Howgrave-Graham, P. Nguyen. Paillier's Cryptosystem Revisited, *ACM Conf. on Comp. and Commun. Security*, 2001, pp. 206–214.
6. Y. Chang. Single Database Private Retrieval with Logarithmic Communication, *ACISP'04*, LNCS 3108, 2004, C. Boyd, J. Gonzalez (Eds.), pp. 50–61.
7. C. Crepeau, J. van de Graaf, A. Tapp. Committed Oblivious Transfer and Private Multi-Party Computation, *CRYPTO'95*, LNCS 963, 1995, D. Coppersmith (Ed.), pp. 110–123.
8. G. Di Crescenzo. Private selective payment protocols. *Financial Crypto'00*, LNCS 1962, 2001, Y. Frankel (Ed.), pp. 72–89.
9. G. Di Crescenzo, R. Ostrovsky, S. Rajagopalan. Conditional Oblivious Transfer and time released encryption, *CRYPTO'99*, LNCS 1592, 1999, J. Stern, (Ed.), pp. 74–89.

10. R. Cramer, I. Damgard. Linear zero-knowledge - a note on efficient zero-knowledge proofs and arguments, *ACM Symp. on Theory of Computing*, 1997, pp. 436–445.
11. R. Cramer, I. Damgard. Secret-key Zero-Knowledge, *TCC'04*, LNCS 2951, 2004, M. Naor (Ed.), pp. 223–237.
12. I. Damgard, N. Fazio, A. Nicolosi. Non-Interactive Zero-Knowledge from Homomorphic Encryption, *TCC'06*, LNCS 3876, 2006, S. Halevi and T. Rabin (Eds.), pp. 41–59.
13. I. Damgard, M. Jurik. A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System, *PKC'01*, LNCS 1992, 2001, K. Kim (Ed.), pp. 119–136.
14. T. ElGamal. A public-key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *CRYPTO'84*, LNCS 196, 1984, G. Blakley, D. Chaum (Eds.), pp. 10–18.
15. A. Fiat, A. Shamir. How to prove yourself: Practical solutions to Identification and Signature Problems, *CRYPTO'86*, LNCS 263, 1987, A. Odlyzko (Ed.) pp. 186–194.
16. M. Freedman, K. Nissim, B. Pinkas. Efficient Private Matching and Set Intersection, *EUROCRYPT'04*, LNCS 3027, 2004, C. Cachin, J. Camenisch (Eds.), pp. 1–19.
17. M. Freedman, Y. Ishai, B. Pinkas, O. Reingold. Keywords Search and Oblivious Pseudorandom Functions, *TCC'05*, LNCS 3378, 2005, J. Kilian (Ed.), pp. 303–324.
18. S. Goldwasser, S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information, *ACM Symp. on Theory of Computing*, 1982, pp. 365–377.
19. B. Goethals, S. Laur, H. Lipmaa, T. Mielikainen. On Private Scalar Product Computation for Privacy-Preserving Data Mining, *ICISC'04*, LNCS 3506, 2004, pp. 104–120.
20. O. Goldreich, S. Micali, A. Wigderson. How to play any mental game. *ACM Symp. on Theory of Computing*, 1987, pp. 218–229.
21. J. Kilian. Founding Cryptography on Oblivious Transfer, *ACM Symp. on Theory of Computing*, 1988, pp. 20–31.
22. H. Lipmaa. Verifiable Homomorphic Oblivious Transfer and Private Equality Test, *ASIACRYPT'03*, LNCS 2894, 2003, C. Lai (Ed.), pp. 416–433.
23. S. Laur, H. Lipmaa. Additive Conditional Disclosure of Secrets and Applications, *Cryptology ePrint Archive: Report 2005/378*.
24. S. Laur, H. Lipmaa, T. Mielikainen. Private Itemset Support Counting, *ICICS'05*, LNCS 3783, 2005, S. Qing et al. (Eds.), pp. 61–71.
25. D. Naccache, J. Stern. A new public-key cryptosystem based on higher residues, *ACM Conf. on Computer and Commun. Security*, 1998, pp. 59–66.
26. M. Naor, B. Pinkas. Oblivious Transfer and Polynomial Evaluation, *ACM STOC*, 1999, pp. 245–254.
27. M. Naor, B. Pinkas. Efficient Oblivious Transfer Protocols, *ACM-SIAM Symp. on Discrete Algorithms*, 2001, pp. 448–457.
28. T. Okamoto, S. Uchiyama. A new public-key cryptosystem as secure as factoring, *EUROCRYPT'98*, LNCS 1403, 1998, K. Nyberg (Ed.), pp. 308–318.
29. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, *EUROCRYPT'99*, LNCS 1592, 1999, J. Stern (Ed.), pp. 223–238.
30. M. Rabin. How to exchange secrets by oblivious transfer, *Technical Report TR-81*, Harvard Aiken Computation Laboratory, 1981.
31. J. Stern. A New and Efficient All-Or-Nothing Disclosure of Secrets Protocol, *ASIACRYPT'98*, LNCS 1514, 1998, K. Ohta, D. Pei (Eds.), pp. 357–371.

Almost Optimum Secret Sharing Schemes Secure Against Cheating for Arbitrary Secret Distribution

Satoshi Obana and Toshinori Araki

NEC Corporation
{obana@bx, t-araki@ek}.jp.nec.com

Abstract. We consider the problem of cheating in secret sharing schemes, cheating in which individuals submit forged shares in the secret reconstruction phase in an effort to make another participant reconstruct an invalid secret. We introduce a novel technique which uses universal hash functions to detect such cheating and propose two efficient secret sharing schemes that employ the functions. The first scheme is nearly optimum with respect to the size of shares; that is, the size of shares is only one bit longer than its existing lower bound. The second scheme possesses a particular merit in that the parameter for the probability of successful cheating can be chosen without regard to the size of the secret. Further, the proposed schemes are proven to be secure regardless of the probability distribution of the secret.

1 Introduction

A secret sharing scheme is a cryptographic primitive in which a secret is divided into shares and distributed among participants in such a way that only a qualified set of participants can recover the secret. It is a fundamental building block for many cryptographic protocols and is often used in the general composition of secure multiparty computations. While seminal papers were presented by Shamir [10] and Blakley [1] more than a quarter century ago, because of its importance in cryptography, it is still being studied actively today.

Tompa and Woll have pointed out that in Shamir's k -out-of- n threshold secret sharing scheme, even a single user can fool other participants by submitting invalid shares at the secret reconstruction phase. They also proposed a scheme which can detect the fact of cheating when invalid shares are submitted at that point. Ogata, Kurosawa and Stinson also have presented an efficient scheme for detecting cheating [8]. While the size of shares in their scheme is proven to be optimum, the scheme is proven to be secure only if the secret is uniformly distributed, and the size of the secret will restrict possible value for the successful cheating probability.

In this paper, we propose two efficient k -out-of- n threshold secret sharing schemes which are secure regardless of the probability distribution of the secret. The first scheme is nearly optimum with respect to the size of shares; that is, the

size of shares is only one bit longer than its existing bound. In the second scheme, the size of shares is somewhat larger than the first scheme, but the second scheme possesses a particular merit in that the successful cheating probability can be chosen without regard to the size of the secret. This is not the case in either the first scheme or the scheme in [8]. The size of shares in the second scheme is much smaller than that in the scheme by Tompa and Woll, which is also secure for arbitrary secret distribution and whose successful cheating probability can be also chosen without regard to the size of the secret. The size of shares in the second scheme will be even smaller than that in [8] when $\epsilon > |\mathcal{S}|^{-1/2}$, where ϵ denotes the successful cheating probability and \mathcal{S} denotes the set of secrets¹. This interesting phenomenon results from inflexibility of parameter values in [8]. Note that the condition $\epsilon > |\mathcal{S}|^{-1/2}$ is quite reasonable since ϵ is usually required to be 2^{-128} or 2^{-256} , whereas the size of the secret can be as large as $|\mathcal{S}| = 2^{1024}$ or more.

The main idea of the proposed schemes is to use universal hash functions (more precisely, a variant of ASU_2 , an *almost strongly universal* class of hash functions) for cheating detection. Here, the key for the universal hash functions is distributedly shared together with the share of the secret. In reconstructing the secret, both the secret and the key are reconstructed, and each participant verifies that the secret and the hash value are consistent. We additionally provide some techniques to reduce the size of shares and to prevent the hash value from revealing any information about the secret.

The rest of the paper is organized as follows. In Section 2, we briefly review models of secret sharing schemes capable of detecting cheating, and we discuss previous works done on them. In Section 3, we introduce a novel technique for detecting cheating via a universal hash family, and we present efficient schemes based on it. In Section 4, we describe two generalizations of the schemes presented in Section 3. In Section 5, we introduce new models which deal with more powerful cheaters than those in existing models, and we present schemes secure in the new models. In Section 6, we summarize our work.

2 Preliminaries

2.1 Secret Sharing Schemes

In secret sharing schemes, there are n participants $\mathcal{P} = \{P_1, \dots, P_n\}$ and a dealer D . The set of participants who are allowed to reconstruct the secret is characterized by an *access structure* $\Gamma \subseteq 2^{\mathcal{P}}$; that is, participants P_{i_1}, \dots, P_{i_k} are allowed to reconstruct the secret if and only if $\{P_{i_1}, \dots, P_{i_k}\} \in \Gamma$ (for instance, the access structure of a k -out-of- n threshold secret sharing scheme is defined by $\Gamma = \{\mathcal{A} \mid \mathcal{A} \in 2^{\mathcal{P}}, |\mathcal{A}| \geq k\}$.) A model consists of two algorithms: **ShareGen** and **Reconst**. Share generation algorithm **ShareGen** takes a secret $s \in \mathcal{S}$ as input and outputs a list (v_1, v_2, \dots, v_n) . Each $v_i \in \mathcal{V}_i$ is called a *share* and is given to a participant P_i . Ordinarily, **ShareGen** is invoked by the dealer. Secret reconstruction algorithm **Reconst** takes a list of shares and outputs a secret $s \in \mathcal{S}$.

¹ Throughout the paper, the cardinality of the set \mathcal{X} is denoted by $|\mathcal{X}|$.

A secret sharing scheme is called *perfect* if the following two conditions are satisfied for the output (v_1, \dots, v_n) of $\text{ShareGen}(\hat{s})$ where the probabilities are taken over the random tape of ShareGen .

1. if $\{P_{i_1}, \dots, P_{i_k}\} \in \Gamma$ then $\Pr[\text{Reconst}(v_{i_1}, \dots, v_{i_k}) = \hat{s}] = 1$,
2. if $\{P_{i_1}, \dots, P_{i_k}\} \notin \Gamma$ then $\Pr[\mathcal{S} = s \mid \mathcal{V}_{i_1} = v_{i_1}, \dots, \mathcal{V}_{i_k} = v_{i_k}] = \Pr[\mathcal{S} = s]$ for any $s \in \mathcal{S}$.

2.2 Secret Sharing Schemes Secure Against Cheating

A secret sharing schemes capable of detecting cheating was first presented by Tompa and Woll [12]. They considered the scenario in which cheaters who do not belong to the access structure submit forged shares in the secret reconstruction phase. Such cheaters will succeed if another participants in the reconstruction accepts an incorrect secret². There are two different models for secret sharing schemes capable of detecting such cheating. Carpentieri, De Santis and Vaccaro [3] first considered a model in which cheaters who *know* the secret try to make another participant reconstruct an invalid secret. We call this model the “*CDV model*.” Recently, Ogata, Kurosawa and Stinson [8] introduced a model with weaker cheaters who *do not* know the secret in forging their shares. We call this model the “*OKS model*.”

As in ordinary secret sharing schemes, each of these models consists of two algorithms. A share generation algorithm ShareGen is the same as that in the ordinary secret sharing schemes. A secret reconstruction algorithm Reconst is slightly changed: it takes a list of shares as input and outputs either a secret or the special symbol \perp ($\perp \notin \mathcal{S}$). Reconst outputs \perp if and only if cheating has been detected. To formalize the models, we define the following simple game for any (k, n) threshold secret sharing scheme $\mathbf{SS} = (\text{ShareGen}, \text{Reconst})$ and for any (not necessarily polynomially bounded) Turing machine $A = (A_1, A_2)$, where A represents cheaters $P_{i_1}, \dots, P_{i_{k-1}}$ who try to cheat P_{i_k} . Please note that in this section and the next we will focus on the (k, n) threshold type access structure. A more general access structure will be discussed in Section 4.

Game(\mathbf{SS}, A)

```

s ← S;      // according to the probability distribution over S.
(v1, ..., vn) ← ShareGen(s);
(i1, ..., ik-1) ← A1(X);
// set X = s for the CDV model, X = ∅ for the OKS model.
(v'i1, ..., v'ik-1, ik) ← A2(vi1, ..., vik-1, X);
    
```

The advantage of cheaters is expressed as $\text{Adv}(\mathbf{SS}, A) = \Pr[s' \in \mathcal{S} \wedge s' \neq s]$, where $s' = \text{Reconst}(v'_{i_1}, v'_{i_2}, \dots, v'_{i_{k-1}}, v_{i_k})$ and the probability is taken over the distribution of \mathcal{S} , and over the random tapes of ShareGen and A .

² Please note that here we focus on the problem of *detecting* the fact of cheating with unconditional security. Neither secret sharing schemes which *identify* cheaters [2,6] nor *verifiable secret sharing schemes* [9,4] are within the scope of this paper.

Definition 1. A (k, n) threshold secret sharing scheme **SS** is called a (k, n, ϵ) -secure secret sharing scheme if $\text{Adv}(\text{SS}, \mathbf{A}) \leq \epsilon$ for any adversary \mathbf{A} .

2.3 Previous Work

In this subsection, we briefly review the known bounds and constructions of (k, n, ϵ) -secure secret sharing schemes. A lower bound for the size of shares in the CDV model is described as follows:

Proposition 1. [3] *In the CDV model, the size of shares for $(k, n, \epsilon_{\text{CDV}})$ -secure secret sharing schemes is lower bounded by $|\mathcal{V}_i| \geq \frac{|S|}{\epsilon_{\text{CDV}}}$.*

Ogata *et al.* improved this bound when the secret is uniformly distributed:

Proposition 2. [8] *In the CDV model, if the secret is uniformly distributed, then the size of shares $|\mathcal{V}_i|$ for $(k, n, \epsilon_{\text{CDV}})$ -secure secret sharing schemes is lower bounded by $|\mathcal{V}_i| \geq \frac{|S|-1}{\epsilon_{\text{CDV}}^2} + 1$.*

Ogata *et al.* also presented the lower bound for the size of shares for $(k, n, \epsilon_{\text{OKS}})$ -secure secret sharing scheme in the OKS model as follows.

Proposition 3. [8] *In the OKS model, the size of shares for $(k, n, \epsilon_{\text{OKS}})$ -secure secret sharing schemes is lower bounded by $|\mathcal{V}_i| \geq \frac{|S|-1}{\epsilon_{\text{OKS}}} + 1$.*

The following corollary may be seen to be straightforward from Proposition 2 since it has to hold for a uniformly distributed secret.

Corollary 1. *In the CDV model, the size of shares for $(k, n, \epsilon_{\text{CDV}})$ -secure secret sharing schemes which satisfy the following two conditions is lower bounded by $|\mathcal{V}_i| \geq \frac{|S|-1}{\epsilon_{\text{CDV}}^2} + 1$. (1) Successful cheating probability is upper bounded by ϵ regardless of the probability distribution of the secret. (2) Share generation is independent of the secret distribution (i.e. ShareGen does not need to know the secret distribution.)*

Because it is in general difficult to determine exact probability distributions, we do not consider here situations in which the share generation algorithm *knows* the secret distribution and shares are generated according to the distribution³.

Within the OKS model, Ogata *et al.* have proposed an elegant $(k, n, \epsilon_{\text{OKS}})$ -secure secret sharing schemes that satisfies the bound of Proposition 3 with equality [8]. The construction is summarized by the following proposition (please refer to [7] for the definition of *difference set*.)

Proposition 4. [8] *If there exists an (N, ℓ, λ) difference set then there exists a $(k, n, \epsilon_{\text{OKS}})$ -secure secret sharing scheme in the OKS model which satisfies the lower bound of Proposition 3 with equality. The scheme is secure if the secret is uniformly distributed.*

³ As mentioned in [8], an example exists in which the size of shares is smaller than the bound of Proposition 2 when the secret is not uniformly distributed and shares are generated according to the distribution.

However, there are two drawbacks in the scheme of [8]. The first is that the scheme is proven to be secure only if the secret is uniformly distributed. This drawback comes from the property of the scheme that the share of the target participant can be uniquely determined from the shares of $k - 1$ cheaters and the secret. Therefore, if there exists a secret which occurs with high probability then cheaters can guess the share of the target participant also with high probability, which causes the successful cheating probability larger than what is expected when the secret is uniformly distributed. The second drawback is that the successful cheating probability is uniquely determined from the size of the secret; that is, ϵ_{OKS} is determined to be $\epsilon_{OKS} = 1/|\mathcal{S}|$ in [8]. On the other hand, the scheme by Tompa and Woll [12] which is secure in the CDV model is proven to be secure for arbitrary secret distribution and the successful cheating probability can be chosen without regard to the size of the secret. However, the size of shares is as large as $|\mathcal{V}_i| = (\frac{(|\mathcal{S}|-1)(k-1)}{\epsilon_{CDV}} + k)^2$.

3 Proposed Schemes

In this section, we propose two efficient (k, n, ϵ_{CDV}) -secure secret sharing schemes in the CDV model which are proven to be secure for any secret distribution. The first scheme is nearly optimum with respect to the size of shares; that is, the size of shares is $|\mathcal{V}_i| = |\mathcal{S}|/\epsilon_{CDV}^2$ which is only one bit longer than the bound of Corollary 1. The size of shares in the second scheme is $|\mathcal{V}_i| = |\mathcal{S}|(\log |\mathcal{S}|)^2/\epsilon_{CDV}^2$. Though the size of share is larger than the first scheme, the second scheme possesses a particular merit in that the size of the secret and the successful cheating probability can be chosen independently.

The underlying (and yet naive) idea of the schemes is to use *almost strongly universal hash functions* ϵ_{CDV} -ASU₂ for cheating detection. A family of hash functions $H : \mathcal{A} \rightarrow \mathcal{B}$ with the properties (1) and (2) below is called an ϵ -ASU₂. (1) For any $x \in \mathcal{A}$ and $y \in \mathcal{B}$, $|\{h_e \in H \mid h_e(x) = y\}| = |H|/|\mathcal{B}|$. (2) For any $x_1, x_2 (\neq x_1) \in \mathcal{A}$ and $y_1, y_2 \in \mathcal{B}$, $|\{h_e \in H \mid h_e(x_1) = y_1, h_e(x_2) = y_2\}| = \epsilon|H|/|\mathcal{B}|$, where h_e denotes the element of H indexed by the *key* $e \in \mathcal{E}$ (clearly $|H| = |\mathcal{E}|$ holds.)

Now, consider the secret sharing scheme in which a randomly chosen key $e \in \mathcal{E}$ of H (where $H : \mathcal{S} \rightarrow \mathcal{B}$ is ϵ_{CDV} -ASU₂) is shared as well as the secret $s \in \mathcal{S}$ using the Shamir's (k, n) threshold secret sharing scheme and hash value $b = h_e(s)$ is open to the public. In the reconstruction phase, a secret \hat{s} and a key \hat{e} are reconstructed and Reconst outputs \hat{s} as the valid secret if and only if $h_{\hat{e}}(\hat{s}) = b$ holds. Intuitively, the scheme seems to be (k, n, ϵ_{CDV}) -secure in the CDV model since knowledge of the secret s does not help cheaters to compute $\hat{s} (\neq s)$ such that $h_{\hat{e}}(\hat{s}) = b$ with probability better than ϵ_{CDV} .

However, we must be careful about the following problems. The first problem is that the key $\hat{e} \in \mathcal{E}$ reconstructed from the shares is not always same as the original one since cheaters can forge the shares of the key for the hash functions. Therefore, we cannot prove the security of the above scheme directly from the properties of ϵ -ASU₂. The second problem is that public (and unforgeable) storage to store the hash value $b = h_e(s)$ is not always available. If the public

storage is not available then the hash value has to be included in the share of each participant, which makes the size of shares larger. Further, we must ensure that the hash value $b = h_e(s)$ does not reveal any information about the secret since the scheme is no longer perfect if it is not the case. To overcome the first problem, we choose the specific ϵ -ASU₂ which can ensure security even when the key for the hash function is forged⁴. To overcome the second and the third problem, we fix the hash value $b = h_e(s)$ to be the constant (e.g. 0) by which we can eliminate the public storage or additional shares without any loss of security.

We use two families of hash functions to construct the schemes. The first scheme is based on the well known $\frac{1}{p}$ -ASU₂ such that $H = \{h_{e_0, e_1} \mid h_{e_0, e_1}(s) = e_0 - s \cdot e_1, e_i \in GF(p)\}$ (e.g. [11].) The second scheme is generalization of the first scheme and is based on the hash family $H = \{h_{e_0, e_1} \mid h_{e_0, e_1}(s_1, \dots, s_N) = e_0 - \sum_{j=1}^N s_j \cdot e_1^j, e_i \in GF(p)\}$ which is proven to be $\frac{N}{p}$ -ASU₂ [5].

3.1 Almost Optimum Scheme

The share generation algorithm **ShareGen** and the share reconstruction algorithm **Reconst** of the first scheme is described as follows where p is a prime power.

Share Generation: On input a secret $s \in GF(p)$, the share generation algorithm **ShareGen** outputs a list of shares (v_1, \dots, v_n) as follows:

1. Choose random $e_0, e_1 \in GF(p)$ such that $e_0 - s \cdot e_1 = 0$.
2. Generate random polynomials $f_s(x), f_{e_0}(x), f_{e_1}(x) \in GF(p)[X]$ of degree $k - 1$ such that $f_s(0) = s, f_{e_0}(0) = e_0$ and $f_{e_1}(0) = e_1$.
3. Compute $v_i = (f_s(i), f_{e_0}(i), f_{e_1}(i))$ and output (v_1, \dots, v_n) .

Secret Reconstruction and Validity Check: On input a list of k shares $(v_{i_1}, \dots, v_{i_k})$, the secret reconstruction algorithm **Reconst** outputs a secret s or \perp as follows:

1. Reconstruct \hat{s}, \hat{e}_0 and \hat{e}_1 from v_{i_1}, \dots, v_{i_k} using Lagrange interpolation.
2. Output s if $\hat{e}_0 - \hat{s} \cdot \hat{e}_1 = 0$ holds. Otherwise **Reconst** outputs \perp .

The properties of the first scheme is summarized by the following theorem.

Theorem 1. *The scheme of §3.1 is (k, n, ϵ) -secure secret sharing schemes in the CDV model with parameters $|\mathcal{S}| = p, \epsilon = 1/p$ and $|\mathcal{V}_i| = p^3 (= |\mathcal{S}|/\epsilon^2)$. Further, the scheme is secure for arbitrary secret distribution.*

The size of shares in the first scheme is only one bit longer than the lower bound of Proposition 2 since $\frac{|\mathcal{S}|}{\epsilon^2} < 2(\frac{|\mathcal{S}|-1}{\epsilon^2} + 1)$ holds for $|\mathcal{S}| \geq 2$.

3.2 A Scheme with Flexible Parameters

In the first scheme, the successful cheating probability is uniquely determined from the size of the secret. On the other hand, the successful cheating probability can be chosen without regard to the size of the secret in the second scheme. The second scheme can be described as follows.

⁴ Formal requirements for the family of hash functions are given in Section 4.

Share Generation: On input a secret $s = (s_1, \dots, s_N) \in GF(p)^N$, the share generation algorithm *ShareGen* outputs a list of shares (v_1, \dots, v_n) according to the following procedure. Please note that we sometimes regard $s = (s_1, \dots, s_N)$ as an element of $GF(p^N)$ instead of $GF(p)^N$.

1. Choose random $e_0, e_1 \in GF(p)$ such that $e_0 - \sum_{j=1}^N s_j e_1^j = 0$.
2. Generate a random polynomials $f_s(x) \in GF(p^N)[X]$ and $f_{e_0}(x), f_{e_1}(x) \in GF(p)[X]$ of degree $k - 1$ such that $f_s(0) = s, f_{e_0}(0) = e_0$ and $f_{e_1}(0) = e_1$.
3. Compute $v_i = (f_s(i), f_{e_0}(i), f_{e_1}(i))$ and output (v_1, \dots, v_n) .

Secret Reconstruction and Validity Check: On input a list of k shares $(v_{i_1}, \dots, v_{i_k})$, the secret reconstruction algorithm *Reconst* outputs a secret s or \perp as follows:

1. Reconstruct \hat{s}, \hat{e}_0 and \hat{e}_1 from v_{i_1}, \dots, v_{i_k} using Lagrange interpolation.
2. Output s if $\hat{e}_0 - \sum_{j=1}^N \hat{s}_j \hat{e}_1^j = 0$ holds. Otherwise *Reconst* outputs \perp .

The following theorem holds for the second scheme. Note that the successful cheating probability ϵ can be chosen flexibly by choosing the prime power p .

Theorem 2. *The scheme of §3.2 is (k, n, ϵ) -secure secret sharing schemes in the CDV model with parameters $|\mathcal{S}| = p^N, \epsilon = N/p, |\mathcal{V}_i| = p^{N+2} (= |\mathcal{S}|(\log_p |\mathcal{S}|)^2 / \epsilon^2)$. Further, the scheme is secure for arbitrary secret distribution.*

Proof. Without loss of generality, we can assume P_1, \dots, P_{k-1} are cheaters and they try to cheat P_k by forging their shares $v_i = (v_{s,i}, v_{e_0,i}, v_{e_1,i})$ ($1 \leq i \leq k - 1$).

We consider two cases depending on whether the cheaters know the secret. In the first case, suppose that the cheaters *know* the secret. The cheaters obtain the following information about e_0 and e_1 from their shares v_1, \dots, v_{k-1} and the secret $s \in \mathcal{S}$: $e_\ell = L_k v_{e_\ell, k} + \sum_{j=1}^{k-1} L_j v_{e_\ell, j}$ (for $\ell = 0, 1$), $e_0 - \sum_{j=1}^N s_j \cdot e_1^j = 0$ where $v_{e_0, k}$ and $v_{e_1, k}$ are unknown to the cheaters and each L_j is a Lagrange coefficient. For simplicity, we will rewrite e_i by $e_i = L_k v_{e_i, k} + C_i$ (for $i = 0, 1$) where $C_i = \sum_{j=1}^{k-1} L_j v_{e_i, j}$ are known to the cheaters. Then we have

$$L_k v_{e_0, k} + C_0 = \sum_{j=1}^N s_j \cdot (L_k v_{e_1, k} + C_1)^j . \tag{1}$$

Now suppose that the cheaters try to cheat P_k by forging their shares to $v'_i = (v'_{s,i}, v'_{e_0,i}, v'_{e_1,i})$ (for $1 \leq i \leq k - 1$.) They succeed in cheating P_k if $e'_0 - \sum_{j=1}^N s'_j \cdot e_1'^j = 0$ holds where e'_0, e'_1 and $s' (\neq s)$ are computed by $e'_0 = L_k v_{e_0, k} + \sum_{j=1}^{k-1} L_j v'_{e_0, j}, e'_1 = L_k v_{e_1, k} + \sum_{j=1}^{k-1} L_j v'_{e_1, j}$ and $s' = L_k v_{s, k} + \sum_{j=1}^{k-1} L_j v'_{s, j}$. Let $C'_i = \sum_{j=1}^{k-1} L_j v'_{e_i, j}$ (for $i = 0, 1$) then the cheaters succeed in cheating if the following equality holds (please note that the cheater can control the values of C'_0, C'_1 and s' as they want by adjusting their shares.⁵)

$$L_k v_{e_0, k} + C'_0 = \sum_{j=1}^N s'_j \cdot (L_k v_{e_1, k} + C'_1)^j \tag{2}$$

⁵ The cheaters can control s' since they can compute $v_{s, k}$ from their shares and s .

The successful cheating probability ϵ is computed as follows:

$$\epsilon = \Pr[s' \in \mathcal{S} \wedge s' \neq s] = \Pr[\text{eq. (1) and eq. (2) hold} \mid \text{eq. (1) holds}] = N/p.$$

We will show the above equation. The condition “eq. (1) and eq. (2) hold” is equivalent to “eq. (1) and eq. (3) hold” where eq. (3) is described as follows:

$$\sum_{j=1}^N s_j \cdot (L_k v_{e_1,k} + C_1)^j - C_0 = \sum_{j=1}^N s'_j \cdot (L_k v_{e_1,k} + C'_1)^j - C'_0. \quad (3)$$

Now let J be the largest number such that $s_J \neq s'_J$, then eq. (3) can be rewritten as the univariate equation $(s_J - s'_J)L_k^J \cdot v_{e_1,k}^J + \sum_{j=0}^{J-1} a_j \cdot v_{e_1,k}^j = 0$ of degree J with the variable $v_{e_1,k}$ where all the coefficients can be arbitrarily controlled by the cheaters except that $(s_J - s'_J)L_k^J \neq 0$. This equation has at most $J (\leq N)$ roots and for each root $v_{e_1,k}$, there exists a unique $v_{e_0,k}$ that satisfies eq. (1). Since the share generation algorithm ShareGen chooses actual $(v_{e_0,k}, v_{e_1,k})$ uniformly and randomly from the p pairs of $(v_{e_0,k}, v_{e_1,k})$ which satisfy eq. (1), we see that the successful cheating probability of the cheaters is upper bounded by N/p .

Now we consider the second case in which the cheaters *do not* know the secret. In this case the successful cheating probability of the cheaters who forge their shares from $v_i = (v_{s,i}, v_{e_0,i}, v_{e_1,i})$ to $v'_i = (v'_{s,i}, v'_{e_0,i}, v'_{e_1,i})$, where at least one $v'_{s,i}$ must satisfy $v'_{s,i} \neq v_{s,i}$, is computed as follows:

$$\begin{aligned} \epsilon &= \sum_{s \in \mathcal{S}} \Pr[\mathcal{S} = s] \Pr[s' \in \mathcal{S} \wedge s' \neq s] \\ &= \sum_{s \in \mathcal{S}} \Pr[\mathcal{S} = s] \Pr[\text{eq. (1) and eq. (2) hold} \mid \text{eq. (1) holds}] = N/p. \end{aligned}$$

The above equality holds since $\Pr[\text{eq. (1) and eq. (2) hold} \mid \text{eq. (1) holds}] = N/p$ holds for any $s \in \mathcal{S}$. □

Note that the above proof includes the proof for Theorem 2 since the first scheme is achieved by setting $N = 1$ in the second scheme.

4 Generalization

In this section, we present more general results on the access structures and on the class of hash functions used to detect cheating.

Though the schemes presented in Section 3 only deal with (k, n) threshold type access structure, we can show that the proposed technique can be applied to any *linear secret sharing schemes*. A linear secret sharing scheme is a class of secret sharing schemes with the following properties: (1) The secret s is an element of a finite field \mathbb{F} . (2) The shares (v_1, \dots, v_n) are generated by $(v_1, v_2, \dots, v_n) = (s, r_1, \dots, r_{t-1})M$ where M is a fixed $t \times n$ matrix over \mathbb{F} and each $r_i \in \mathbb{F}$ is chosen randomly. (3) For a set of participants $\mathcal{P} = \{P_{i_1}, \dots, P_{i_j}\} \in \Gamma$ and their shares $(v_{i_1}, \dots, v_{i_j})$, the secret s is computed by $s = \sum_{k=1}^j c_{\mathcal{P},j} \cdot v_{i_j}$ where each $c_{\mathcal{P},j} \in \mathbb{F}$ is a constant uniquely determined from \mathcal{P} .

We can also generalize the class of hash function used to detect cheating. To characterize such class of hash function, we define a new class of hash function called *strongly key-differential universal* (ϵ -SKDU₂ for short) as follows:

Definition 2. A family of hash functions $H : \mathcal{A} \rightarrow \mathcal{B}$ is called a strongly key-differential universal ϵ -SKDU₂ if there exists $\hat{b} \in \mathcal{B}$ such that for any distinct $a, a' \in \mathcal{A}$ and for any $c \in \mathcal{E}$,

$$\frac{|\{h_e \mid e \in \mathcal{E}, h_e(a) = \hat{b}, h_{e+c}(a') = \hat{b}\}|}{|\{h_e \mid e \in \mathcal{E}, h_e(a) = \hat{b}\}|} \leq \epsilon. \tag{4}$$

Further, ϵ -SKDU₂ is called an “efficiently samplable” if there exists an efficient (i.e. polynomial time) algorithm to choose $e \in \mathcal{E}$ randomly from the set $\{e \in \mathcal{E} \mid h_e(a) = \hat{b}\}$ for any $a \in \mathcal{A}$.

The following theorem shows that we can construct secret sharing scheme capable of detecting cheating in the CDV model from any linear secret sharing schemes over \mathcal{S} and over \mathcal{E} , and any efficiently samplable ϵ -SKDU₂ with the domain \mathcal{S} .

Theorem 3. If there exist linear secret sharing schemes over \mathcal{S} and \mathcal{E} for a common access structure Γ and an efficiently samplable ϵ -SKDU₂ $H : \mathcal{S} \rightarrow \mathcal{B}$, then there exists a secret sharing scheme capable of detecting cheating for the access structure Γ in the CDV model such that the successful cheating probability is equal to ϵ for arbitrary secret distribution.

Proof. Let \mathcal{S} and \mathcal{E} be a set of the secrets and the set of keys for ϵ -SKDU₂, respectively and let $\mathbf{SS}_1 = (\text{ShareGen}_1, \text{Reconst}_1)$ and $\mathbf{SS}_2 = (\text{ShareGen}_2, \text{Reconst}_2)$ be linear secret sharing schemes over \mathcal{S} and over \mathcal{E} for the same access structure Γ , respectively. We construct a secret sharing scheme secure against cheaters $\mathbf{SS} = (\text{ShareGen}, \text{Reconst})$ as follows.

Share Generation: On input a secret $s \in \mathcal{S}$, the share generation algorithm ShareGen outputs a list of shares (v_1, \dots, v_n) as follows:

1. Choose a random $e \in \mathcal{E}$ such that $h_e(s) = \hat{b}$, which can be computed efficiently since the efficiently samplable ϵ -SKDU₂ is used.
2. Generate $(v_{s,1}, \dots, v_{s,n}) \leftarrow \text{ShareGen}_1(s)$ and $(v_{e,1}, \dots, v_{e,n}) \leftarrow \text{ShareGen}_2(e)$.
3. Compute the share $v_i = (v_{s,i}, v_{e,i})$ of each P_i and output (v_1, \dots, v_n) .

Secret Reconstruction and Validity Check: On input t shares $(v_{i_1}, \dots, v_{i_t})$ such that $\{P_{i_1}, \dots, P_{i_t}\} \in \Gamma$, the secret reconstruction algorithm Reconst outputs a secret $s \in \mathcal{S}$ or \perp as follows:

1. Compute $\hat{s} \leftarrow \text{Reconst}_1(v_{s,i_1}, \dots, v_{s,i_t})$ and $\hat{e} \leftarrow \text{Reconst}_2(v_{e,i_1}, \dots, v_{e,i_t})$.
2. Output s if $h_{\hat{e}}(\hat{s}) = \hat{b}$. Otherwise Reconst outputs \perp .

Now we show that $\mathbf{SS} = (\text{ShareGen}, \text{Reconst})$ constructed above is ϵ -secure. Without loss of generality we can assume that $\mathcal{P} = \{P_1, \dots, P_t\}$ is an element of Γ and that P_1, \dots, P_{t-1} are cheaters who try to cheat P_t . There are two cases to consider. In the first case, suppose that the cheaters *know* the secret.

Let $v_i = (v_{s,i}, v_{e,i})$ be the share of P_i . Since the cheaters know their shares v_1, \dots, v_{t-1} and the secret s and that \mathbf{SS}_1 and \mathbf{SS}_2 are the linear secret sharing

schemes, the cheaters know $h_e(s) = \hat{b}$ holds where e is computed by $e = c_{\mathcal{P},t}v_{e,t} + \sum_{j=1}^{t-1} c_{\mathcal{P},j}v_{e,j}$ for a constant $c_{\mathcal{P},i}$. Now suppose the cheaters try to cheat P_t by forging their shares to $v'_i = (v'_{s,i}, v'_{e,i})$ (for $1 \leq i \leq t - 1$.) They succeed in cheating P_t if $h_{e'}(s') = \hat{b}$ holds for e' and $s' (\neq s)$ computed by $e' = c_{\mathcal{P},t}v_{e',t} + \sum_{j=1}^{t-1} c_{\mathcal{P},j}v'_{e',j}$, $s' = c_{\mathcal{P},t}v_{s,t} + \sum_{j=1}^{t-1} c_{\mathcal{P},j}v'_{s,j}$. Since $e' = e + \sum_{j=1}^{t-1} c_{\mathcal{P},j}(v'_{e',j} - v_{e,j})$ holds, we see that the cheaters succeed in cheating if $h_{e+C}(s') = \hat{b}$ holds where $C = \sum_{j=1}^{t-1} c_{\mathcal{P},j}(v'_{e',j} - v_{e,j})$ is known to the cheaters. Therefore, the successful cheating probability ϵ is computed as follows.

$$\begin{aligned} & \Pr[s' \in \mathcal{S} \wedge s' \neq s] \\ &= \Pr[h_e(s) = \hat{b} \text{ and } h_{e+C}(s') = \hat{b} \mid h_e(s) = \hat{b}] \\ &= \frac{\Pr[h_e(s) = \hat{b} \text{ and } h_{e+C}(s') = \hat{b}]}{\Pr[h_e(s) = \hat{b}]} = \frac{|\{h_e \mid h_e(s) = \hat{b}, h_{e+C}(s') = \hat{b}\}|}{|\{h_e \mid h_e(s) = \hat{b}\}|} \leq \epsilon \end{aligned}$$

where the last equation directly follows from eq. (4).

It can be proven that the successful cheating probability is upper bounded by ϵ when the cheaters *do not* know the secret by the same technique used in Theorem 2. □

It is easily checked that the families of hash function used in the proposed schemes of Section 3 meet the requirements of efficiently samplable ϵ -SKDU₂.

Constructions of ϵ -SKDU₂ other than those used in the proposed schemes will be of independent interest. The following theorem shows that an ϵ -SKDU₂ (and therefore, a secret sharing scheme capable of detecting cheating) can be constructed from an ϵ -ASU₂ with additional properties.

Theorem 4. *If a family of hash functions $H : \mathcal{A} \rightarrow \mathcal{B}$ is an ϵ -ASU₂ with the properties (1) and (2) below then H is an efficiently samplable ϵ -SKDU₂.*

(1) *H is constructed from $H_\Delta : \mathcal{A} \rightarrow \mathcal{B}$ of ϵ -A Δ U₂ as follows, where ϵ -A Δ U₂ is a family of hash functions such that $|\{h_e \in H_\Delta \mid h_e(a) - h_e(a') = b\}| = \epsilon|H|$ for any distinct $a, a' \in \mathcal{A}$ and for any $b \in \mathcal{B}$.*

$$H = \{h_{e_0, e_1} \mid h_{e_0, e_1}(a) = h'_{e_0}(a) + e_1, h'_{e_0} \in H_\Delta, e_1 \in \mathcal{B}\}$$

(2) *H_Δ is linear with respect to the key; that is, $h'_{e+e'}(a) = h'_e(a) + h'_{e'}(a)$ holds for any $e, e' \in \mathcal{E}$ and for any $a \in \mathcal{A}$.*

Proof. It is well known that the family of hash functions H constructed as above is ϵ -ASU₂ (please refer to [11] for the proof.) Let \hat{b} be an arbitrary element of \mathcal{B} then we will show that H satisfies the conditions of an efficiently samplable ϵ -SKDU₂. First, it is easy to see that e_0 and e_1 such that $h_{e_0, e_1}(a) = \hat{b}$ is efficiently samplable by choosing $e_0 \in \mathcal{E}$ randomly and by computing $e_1 = \hat{b} - h_{e_0}(a)$. Next, we show that eq. (4) holds for H . Since H is constructed based on H_Δ with the property $h'_{e+e'}(a) = h'_e(a) + h'_{e'}(a)$ for any $h' \in H_\Delta$, $h_{e_0+c_0, e_1+c_1}(a) = h'_{e_0+c_0}(a) + (e_1 + c_1) = (h'_{e_0}(a) + e_1) + (h'_{c_0}(a) + c_1) = h_{e_0, e_1}(a) + h_{c_0, c_1}(a)$ holds

for any $a \in \mathcal{A}$ and for any $(e_0, e_1), (c_0, c_1) \in \mathcal{E} \times \mathcal{B}$. Therefore, the following equation holds.

$$\begin{aligned} & |\{h_{e_0, e_1} \in H \mid h_{e_0, e_1}(a) = \hat{b}, h_{e_0+c_0, e_1+c_1}(a') = \hat{b}\}| \\ &= |\{h_{e_0, e_1} \in H \mid h_{e_0, e_1}(a) = \hat{b}, h_{e_0, e_1}(a') = \hat{b} - h_{c_0, c_1}(a')\}| \\ &= |\{h_{e_0, e_1} \in H \mid h_{e_0, e_1}(a) = \hat{b}, h_{e_0, e_1}(a') = \hat{b}'\}| = \epsilon|H|/|\mathcal{B}| \end{aligned}$$

where the last equation follows from the second condition of ϵ -ASU₂. Combining the above equation and the first property of ϵ -ASU₂: $|\{h_{e_0, e_1} \in H \mid h_{e_0, e_1}(a) = \hat{b}\}| = |H|/|\mathcal{B}|$, we have $\frac{|\{h_{e_0, e_1} \in H \mid h_{e_0, e_1}(a) = \hat{b}, h_{e_0+c_0, e_1+c_1}(a') = \hat{b}\}|}{|\{h_{e_0, e_1} \in H \mid h_{e_0, e_1}(a) = \hat{b}\}|} = \epsilon$ for any distinct $a, a' \in \mathcal{A}$ and for any $(c_0, c_1) \in \mathcal{E} \times \mathcal{B}$. □

Please note that the family of hash function used in the first scheme is constructed based on Theorem 4, whereas the family of hash function used in the second scheme is not. Therefore, we see that SKDU₂ can be constructed by other means than Theorem 4.

5 Coping with More Powerful Cheaters

In this section, we consider the models with more powerful cheaters than those in the OKS and the CDV models and we present secure schemes against them.

In the OKS model and the CDV model, the secret reconstruction algorithm *Reconst* is defined to take only a list of share $(v_{i_1}, \dots, v_{i_k})$ as input. In actual schemes, however, the identities of the owners i_1, \dots, i_k are usually required to reconstruct the secret. This means that we implicitly assume there exist means to know the *correct* identities of share holders in the secret reconstruction phase of both the OKS and the CDV models. In the real life, however, it is very difficult to realize an identification scheme secure against adversaries with unlimited computational power. Therefore, it is highly desired to construct secret sharing schemes capable of detecting cheating without relying on secure identification.

To this end, we define new models: the OKS⁺ model and the CDV⁺ model which are slight modifications of the OKS model and the CDV model, respectively. In both new models, we modify a secret reconstruction algorithm *Reconst* and a game *Game*⁺ of cheaters $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ against $\mathbf{SS} = (\text{ShareGen}, \text{Reconst})$ as follows. The secret reconstruction algorithm *Reconst* takes a list $((i_1, v_{i_1}), (i_2, v_{i_2}), \dots, (i_k, v_{i_k}))$ of pairs of an identity i_ℓ and a share v_{i_ℓ} of P_{i_ℓ} . Cheaters in the new models are allowed to forge their identities as well as their shares. To characterize such cheaters, a game *Game*⁺ is defined as follows.

Game⁺(\mathbf{SS}, \mathbf{A})

```

 $s \leftarrow \mathcal{S};$  // according to the probability distribution over  $\mathcal{S}$ .
 $(v_1, \dots, v_n) \leftarrow \text{ShareGen}(s);$ 
 $(i_i, \dots, i_{k-1}) \leftarrow \mathbf{A}_1(X);$ 
// set  $X = s$  for the CDV+ model,  $X = \emptyset$  for the OKS+ model.
 $((i'_1, v'_{i'_1}), \dots, (i'_{k-1}, v'_{i'_{k-1}}), i_k) \leftarrow \mathbf{A}_2(v_{i_1}, \dots, v_{i_{k-1}}, X);$ 

```

The advantage of cheaters is redefined by $Adv(\mathbf{SS}, \mathbf{A}) = \Pr[s' \in \mathcal{S} \wedge s' \neq s]$, where $s' = \text{Reconst}((i'_1, v'_{i'_1}), (i'_2, v'_{i'_2}), \dots, (i'_{k-1}, v'_{i'_{k-1}}), (i_k, v_{i_k}))$ and the probability is taken over the distribution of \mathcal{S} , and over the random tapes of ShareGen and \mathbf{A} .

Note that all the bounds for the OKS model (resp., the CDV model) (e.g. Propositions 1–3 and Corollary 1) are also valid for OKS⁺ model (resp., the CDV⁺ model) since a scheme secure in the OKS⁺ model (resp., the CDV⁺ model) are also secure in the OKS model (resp., the CDV model).

Though the schemes secure in the OKS model (resp., the CDV model) are not necessarily secure in the OKS⁺ model (resp., the CDV⁺ model,) the scheme presented in [8] can be proven to be secure in the OKS⁺ model and the scheme presented in [12] can be proven to be secure in the CDV⁺ model. With respect to the proposed schemes, the first scheme can be shown to be secure in the CDV⁺ model. However, the second scheme is *not* secure in the CDV⁺ model. This is because the security proof of the second scheme strongly relies on the fact that the cheaters can not manipulate the Lagrange coefficient L_k , which is not the case in the CDV⁺ model. When cheaters can manipulate the Lagrange coefficient as they want, they will succeed in cheating with probability one, which is possible by forging the Lagrange coefficient L_k to $L'_k (\neq L_k)$ in eq. (2) and by adjusting s'_j, C'_0 and C'_1 to make eq. (2) equivalent to eq. (1).

The good news is that the second scheme secure can be made secure in CDV⁺ model by slight modification. The main idea of the modified scheme is to introduce a *constant padding* to a hash function. Specifically, we choose a key e of a hash families with which $h_e(s_1, \dots, s_N, 1, 1, 0, 1) = 0$ instead of choosing a key such that $h_e(s_1, \dots, s_N) = 0$ as in the second scheme. In this modified scheme, we can show that cheaters cannot make eq. (2) equivalent to eq. (1) unless they leave the Lagrange coefficient L_k and the secret $s = (s_1, \dots, s_N)$ unchanged. The modified scheme can be described as follows.

Share Generation: On input a secret $s = (s_1, \dots, s_N) \in GF(p)^N$, the share generation algorithm ShareGen outputs a list of shares (v_1, \dots, v_n) according to the following procedure. Please note that we sometimes regard $s = (s_1, \dots, s_N)$ as an element of $GF(p^N)$ instead of $GF(p)^N$.

1. Choose random $e_0, e_1 \in GF(p)$ such that $e_0 - (e_1^{N+4} + e_1^{N+2} + e_1^{N+1} + \sum_{j=1}^N s_j e_1^j) = 0$.
2. Generate random polynomials $f_s(x) \in GF(p^N)[X]$ and $f_{e_0}(x), f_{e_1}(x) \in GF(p)[X]$ of degree $k - 1$ such that $f_s(0) = s, f_{e_0}(0) = e_0$ and $f_{e_1}(0) = e_1$.
3. Compute $v_i = (f_s(i), f_{e_0}(i), f_{e_1}(i))$ and output (v_1, \dots, v_n) .

Secret Reconstruction and Validity Check: On input a list of k pair of identities and shares $((i_1, v_{i_1}), \dots, (i_k, v_{i_k}))$, the secret reconstruction algorithm Reconst outputs a secret s or \perp according to the following procedure.

1. Reconstruct \hat{s}, \hat{e}_0 and \hat{e}_1 from v_{i_1}, \dots, v_{i_k} using Lagrange interpolation.
2. Output s if $\hat{e}_0 - (\hat{e}_1^{N+4} + \hat{e}_1^{N+2} + \hat{e}_1^{N+1} + \sum_{j=1}^N \hat{s}_j \hat{e}_1^j) = 0$ holds. Otherwise Reconst outputs \perp .

Security of the modified scheme is summarized by the following theorem.

Theorem 5. *The modified scheme presented above is (k, n, ϵ) -secure secret sharing schemes in the CDV^+ model with the following parameters: $|\mathcal{S}| = p^N, \epsilon = (N+4)/p$ and $|\mathcal{V}_i| = p^{N+2} (= |\mathcal{S}|(\log_p |\mathcal{S}|+4)^2/\epsilon^2)$. Further, the scheme is secure for arbitrary secret distribution.*

Proof. The proof is similar to that of Theorem 2. Let P_j ($1 \leq j \leq k-1$) be cheaters who try to cheat P_k by forging their identities j to $i_j (\neq k)$ and corresponding shares to $v'_{i_j} = (v'_{s,i_j}, v'_{e_0,i_j}, v'_{e_1,i_j})$ ($1 \leq j \leq k-1$).

As in the proof of Theorem 2, we consider two cases depending on whether the cheaters know the secret. In the first case, suppose that the cheaters *know* the secret. The cheaters obtain the following information about e_0 and e_1 from their shares v_1, \dots, v_{k-1} and the secret $s \in \mathcal{S}$: $e_\ell = L_k v_{e_\ell,k} + \sum_{j=1}^{k-1} L_j v_{e_\ell,j}$ ($\ell = 0, 1$), $e_0 - (e_1^{N+4} + e_1^{N+2} + e_1^{N+1} + \sum_{j=1}^N s_j \cdot e_1^j) = 0$ where $v_{e_0,k}$ and $v_{e_1,k}$ are unknown to the cheaters and each L_j is a Lagrange coefficient. For simplicity, we will rewrite e_i by $e_i = L_k v_{e_i,k} + C_i$ (for $i = 0, 1$) where $C_i = \sum_{j=1}^{k-1} L_j v_{e_i,j}$ is known to the cheaters. Then we have the following equality.

$$L_k v_{e_0,k} + C_0 = \sum_{j \in \{1,2,4\}} (L_k v_{e_1,k} + C_1)^{N+j} + \sum_{j=1}^N s_j \cdot (L_k v_{e_1,k} + C_1)^j \quad (5)$$

Now suppose the cheaters P_j ($1 \leq j \leq k-1$) try to cheat P_k by forging their identities to i_j and by forging corresponding shares to $v'_{i_j} = (v'_{s,i_j}, v'_{e_0,i_j}, v'_{e_1,i_j})$. They succeed in cheating P_k if $e'_0 - (\sum_{j \in \{1,2,4\}} e_1'^{N+j} + \sum_{j=1}^N s'_j \cdot e_1'^j) = 0$ holds where e'_0, e'_1 and $s' (\neq s)$ are computed by $e'_\ell = L'_k v_{e_\ell,k} + \sum_{j=1}^{k-1} L'_j v'_{e_\ell,i_j}$ (for $\ell = 0, 1$), $s' = L'_k v_{s,k} + \sum_{j=1}^{k-1} L'_j v'_{s,i_j}$. Let $C'_\ell = \sum_{j=1}^{k-1} L'_j v'_{e_\ell,i_j}$ (for $\ell = 0, 1$) then the cheaters succeed in cheating if the following equality holds (as in Theorem 2, the cheaters can control the values of C'_0, C'_1 and s' as they want.)

$$L'_k v_{e_0,k} + C'_0 = \sum_{j \in \{1,2,4\}} (L'_k v_{e_1,k} + C'_1)^{N+j} + \sum_{j=1}^N s'_j \cdot (L'_k v_{e_1,k} + C'_1)^j \quad (6)$$

The successful cheating probability ϵ is computed by $\epsilon = \Pr[s' \in \mathcal{S} \wedge s' \neq s] = \Pr[\text{eq. (5) and eq. (6) hold} \mid \text{eq. (5) holds}]$. We will show that $\epsilon = (N+4)/p$. First, assume that eq. (5) is not equivalent to eq. (6) (i.e. $L'_k \times \text{eq. (5)}$ is not identical to $L_k \times \text{eq. (6)}$.) In this case, ϵ is proven to be $(N+4)/p$ by similar discussion to the proof of Theorem 2. Next, we will show that if the cheaters make eq. (6) equivalent to eq. (5) then successful cheating probability becomes 0. This can be proven by showing that eq. (5) is equivalent to eq. (6) only if the $L'_k = L_k, C'_i = C_i$ (for $i = 0, 1$) and $s_j = s'_j$ (for $1 \leq j \leq N$) since the cheaters succeed in cheating only when P_k accepts s' such that $s' \neq s$. Suppose $L_k \times \text{eq. (5)}$ and $L'_k \times \text{eq. (6)}$ are identical then their coefficients of $v_k^{N+4}, v_k^{N+3}, v_k^{N+2}$ and v_k^{N+1} must be identical. Therefore, we have the following equations.

$$L'_k L_k^{N+4} = L_k L_k^{N+4} \quad (7)$$

$$\binom{N+4}{1} C_1 L'_k L_k^{N+3} = \binom{N+4}{1} C'_1 L_k L_k^{N+3} \quad (8)$$

$$\left(\binom{N+4}{2} C_1^2 + 1 \right) L'_k L_k^{N+2} = \left(\binom{N+4}{2} C_1'^2 + 1 \right) L_k L_k'^{N+2} \tag{9}$$

$$\left(\binom{N+4}{3} C_1^3 + \binom{N+2}{1} C_1 + 1 \right) L'_k L_k^{N+1} = \left(\binom{N+4}{3} C_1'^3 + \binom{N+2}{1} C_1' + 1 \right) L_k L_k'^{N+1} \tag{10}$$

From eq. (7) and eq. (8) we have $L_k^{N+3} = L_k'^{N+3}$ and $C_1/L_k = C_1'/L'_k$. Using these relations eq. (7)–eq. (10) can be rewritten as follows.

$$L_k^{N+3} = L_k'^{N+3}, \quad C_1/L_k = C_1'/L'_k, \quad L_k^{N+1} = L_k'^{N+1}, \quad L_k^N = L_k'^N$$

The above equalities holds if and only if $L_k = L'_k$ and $C_1 = C_1'$. Further, $s_j = s'_j$ (for $1 \leq j \leq N$) can be also derived from the condition that the coefficients of v_k^j in eq. (5) and eq. (6) are identical. Finally, $C_0 = C_0'$ is derived from the condition that the constant terms of eq. (5) and eq. (6) are identical.

Now we consider the second case in which the cheaters *do not* know the secret. In this case the successful cheating probability of the cheaters who forge their identities and corresponding shares from $(j, (v_{s,j}, v_{e_0,j}, v_{e_1,j}))$ to $(i_j, (v'_{s,i_j}, v'_{e_0,i_j}, v'_{e_1,i_j}))$ is computed as follows:

$$\begin{aligned} \epsilon &= \sum_{s \in \mathcal{S}} \Pr[\mathcal{S} = s] \Pr[s' \in \mathcal{S} \wedge s' \neq s] \\ &= \sum_{s \in \mathcal{S}} \Pr[\mathcal{S} = s] \Pr[\text{eq. (5) and eq. (6) hold} \mid \text{eq. (5) holds}] = (N + 4)/p. \end{aligned}$$

The above equality holds since $\Pr[\text{eq. (5) and eq. (6) hold} \mid \text{eq. (5) holds}] = (N + 4)/p$ holds for any $s \in \mathcal{S}$. □

The following theorem gives a generalized result analogous to Theorem 3.

Theorem 6. *If there exist linear secret sharing schemes over \mathcal{S} and \mathcal{E} for a common access structure Γ and a family of hash functions $H : \mathcal{S} \rightarrow \mathcal{B}$ which satisfies the conditions (1)–(3) below, then there exists a secret sharing scheme capable of detecting cheating for the access structure Γ in the CDV⁺ model such that the successful cheating probability equals ϵ for arbitrary secret distribution.*

1. Addition and (scalar) multiplication over the set of keys \mathcal{E} of H are defined.
2. There exists $\hat{b} \in \mathcal{B}$ such that for any distinct $a, a' \in \mathcal{A}$ and for any c_0 and $c_1 \in \mathcal{E}$, $\frac{|\{h_e \mid e \in \mathcal{E}, h_e(a) = \hat{b}, h_{c_0e+c_1}(a') = \hat{b}\}|}{|\{h_e \mid e \in \mathcal{E}, h_e(a) = \hat{b}\}|} \leq \epsilon$. holds.
3. There exists an efficient (i.e. polynomial time) algorithm to choose $e \in \mathcal{E}$ randomly from the set $\{e \in \mathcal{E} \mid h_e(a) = \hat{b}\}$ for any $a \in \mathcal{A}$.

Proof. The proof is similar to that of Theorem 3. Let \mathcal{S} and \mathcal{E} be a set of the secrets and the set of keys for a function family H , respectively. Further, let $\mathbf{SS}_1 = (\text{ShareGen}_1, \text{Reconst}_1)$ and $\mathbf{SS}_2 = (\text{ShareGen}_2, \text{Reconst}_2)$ be linear secret sharing schemes over \mathcal{S} and over \mathcal{E} for the same access structure Γ , respectively. The share generation algorithm ShareGen and Reconst are identical to those defined in the proof of Theorem 3 except that the family of hash functions used here meets the condition 1–3 of Theorem 6.

Now we show that the above **SS** = (ShareGen, Reconst) is ϵ -secure even when the cheaters forge their identities as well as their shares. Without loss of generality we can assume that $\mathcal{P} = \{P_1, \dots, P_t\}$ is an element of Γ and that P_1, \dots, P_{t-1} are cheaters who try to cheat P_t . There are two cases to consider. In the first case, suppose that the cheaters *know* the secret. Let $v_i = (v_{s,i}, v_{e,i})$ be the share of P_i . Since the cheaters know their shares v_1, \dots, v_{t-1} and the secret s and that **SS**₁ and **SS**₂ are the linear secret sharing schemes, the cheaters know $h_e(s) = \hat{b}$ holds where e is computed by $e = c_{\mathcal{P},t}v_{e,t} + \sum_{j=1}^{t-1} c_{\mathcal{P},j}v_{e,j}$ for a constant $c_{\mathcal{P},i}$. Now suppose the cheaters try to cheat P_t by forging their identities from j to i_j (for $1 \leq j \leq t-1$) and corresponding shares to $v'_{i_j} = (v'_{s,i_j}, v'_{e,i_j})$ (for $1 \leq j \leq t-1$.) They succeed in cheating P_t if $h_{e'}(s') = \hat{b}$ holds for e' and $s' (\neq s)$ computed by $e' = c'_{\mathcal{P},t}v_{e,t} + \sum_{j=1}^{t-1} c'_{\mathcal{P},i_j}v'_{e,i_j}$, $s' = c'_{\mathcal{P},t}v_{s,t} + \sum_{j=1}^{t-1} c'_{\mathcal{P},i_j}v'_{s,i_j}$. Since $e' = (\frac{c'_{\mathcal{P},t}}{c_{\mathcal{P},t}})e + \sum_{j=1}^{t-1} (c'_{\mathcal{P},i_j}v'_{e,i_j} - \frac{c_{\mathcal{P},t}c'_{\mathcal{P},i_j}}{c'_{\mathcal{P},t}} \cdot v_{e,j})$ holds, we see that the cheaters succeed in cheating if $h_{C_0 \cdot e + C_1}(s') = \hat{b}$ holds where $C_0 = c'_{\mathcal{P},t}/c_{\mathcal{P},t}$ and $C_1 = \sum_{j=1}^{t-1} (c'_{\mathcal{P},i_j}v'_{e,i_j} - \frac{c_{\mathcal{P},t}c'_{\mathcal{P},i_j}}{c'_{\mathcal{P},t}} \cdot v_{e,j})$ are known to the cheaters. Therefore, the successful cheating probability ϵ is computed as follows.

$$\begin{aligned} \Pr[s' \in \mathcal{S} \wedge s' \neq s] &= \Pr[h_e(s) = \hat{b} \text{ and } h_{C_0 \cdot e + C_1}(s') = \hat{b} \mid h_e(s) = \hat{b}] \\ &= \frac{|\{h_e \mid h_e(s) = \hat{b}, h_{C_0 \cdot e + C_1}(s') = \hat{b}\}|}{|\{h_e \mid h_e(s) = \hat{b}\}|} \leq \epsilon \end{aligned}$$

where the last equation directly follows from the condition (2) of Theorem 6.

It can be proven that the successful cheating probability is upper bounded by ϵ when the cheaters *do not* know the secret by the same technique used in Theorem 5. □

6 Conclusion

In this paper, we proposed two efficient $(k, n, \epsilon_{\text{CDV}})$ -secure secret sharing schemes in the CDV model which are proven to be secure for arbitrary secret distribution. The first scheme is nearly optimum with respect to the size of shares; that is, the size of share is only one bit longer than the lower bound of Corollary 1. In the second scheme, the size of share is larger than that in the first scheme. However, the second scheme possesses a particular merit in that the successful cheating probability can be chosen without regard to the size of the secret. Table 1 below compares the bit length of shares in the three schemes for the various security parameters where the secret is 1024 bit and the access structure considered is 3-out-of-5 threshold type access structure. Compared to the scheme of [12] the size of shares in the proposed scheme (the second scheme) is smaller for all security parameters. It is interesting to note that, when $\epsilon > |\mathcal{S}|^{-1/2}$, the size of the share in the proposed scheme is even smaller than that in [8] which is proven to be secure only in the OKS model. This is because ϵ is determined to be $\epsilon = 2^{-1024}$ when the secret is 1024 bit in the scheme of [8]. Therefore, ϵ is forced to be 2^{-1024}

Table 1. Comparison table of the bit length of the shares (for the secret of 1024 bit)

ϵ	Proposed Scheme	Tompa and Woll	Ogata <i>et al.</i>
2^{-128}	1286	2306	2048
2^{-256}	1540	2562	2048
2^{-512}	2050	3074	2048
2^{-1024}	3072	4098	2048

in [8] even when we only require the security level of $\epsilon = 2^{-128}$ or $\epsilon = 2^{-256}$, which makes the size of share larger than that in the proposed scheme when ϵ is relatively large (please note that $\epsilon = 2^{-128}$ or $\epsilon = 2^{-256}$ will be secure enough in most settings.)

It will be a future study to find $(k, n, \epsilon_{\text{CDV}})$ -secure secret sharing schemes in the CDV model which are secure for arbitrary secret distribution and the bound of Corollary 1 is satisfied with equality.

References

1. G. R. Blakley, "Safeguarding cryptographic keys," Proc. AFIPS 1979, National Computer Conference, vol. 48, pp. 313–137, 1979. vol. 4, no. 4, pp. 502–510, 1991.
2. M. Carpentieri, "A Perfect Threshold Secret Sharing Scheme to Identify Cheaters," Designs, Codes and Cryptography, vol. 5, no. 3, pp. 183–187, 1995.
3. M. Carpentieri, A. De Santis and U. Vaccaro, "Size of Shares and Probability of Cheating in Threshold Schemes," Proc. Eurocrypt'93, Lecture Notes in Computer Science, vol. 765, Springer Verlag, pp. 118–125, 1993.
4. R. Cramer, I. Damgård and U. M. Maurer, "General Secure Multi-party Computation from any Linear Secret-Sharing Scheme," Proc. Eurocrypt'00, Lecture Notes in Computer Science, vol. 1807, Springer Verlag, pp. 316–334, 2000.
5. B. den Boer, "A Simple and Key-Economical Unconditional Authentication Scheme," Journal of Computer Security, vol. 2, pp. 65–71, 1993.
6. K. Kurosawa, S. Obana and W. Ogata, " t -Cheater Identifiable (k, n) Secret Sharing Schemes," Proc. Crypto'95, Lecture Notes in Computer Science, vol. 963, Springer Verlag, pp. 410–423, 1995.
7. F. MacWilliams and N. Sloane, "The Theory of Error Correcting Codes," North Holland, Amsterdam, 1977.
8. W. Ogata, K. Kurosawa and D. R. Stinson, "Optimum Secret Sharing Scheme Secure against Cheating," SIAM Journal on Discrete Mathematics, vol. 20, no. 1, pp. 79–95, 2006.
9. T. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," Proc. Crypto'91, Lecture Notes in Computer Science, vol 576, Springer Verlag, pp. 129–149, 1991.
10. A. Shamir, "How to Share a Secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, 1979.
11. D. R. Stinson, "On the Connections between Universal Hashing, Combinatorial Designs and Error-Correcting Codes," Congressus Numerantium 114, pp. 7–27, 1996.
12. M. Tompa and H. Woll, "How to Share a Secret with Cheaters," Journal of Cryptology, vol. 1, no. 3, pp. 133–138, 1989.

KFC - The Crazy Feistel Cipher

Thomas Baignères* and Matthieu Finiasz

EPFL

CH-1015 Lausanne – Switzerland

<http://lasecwww.epfl.ch>

Abstract. We introduce KFC, a block cipher based on a three round Feistel scheme. Each of the three round functions has an SPN-like structure for which we can either compute or bound the advantage of the best d -limited adaptive distinguisher, for any value of d . Using results from the decorrelation theory, we extend these results to the whole KFC construction. To the best of our knowledge, KFC is the first practical (in the sense that it can be implemented) block cipher to propose tight security proofs of resistance against large classes of attacks, including most classical cryptanalysis (such as linear and differential cryptanalysis, taking hull effect in consideration in both cases, higher order differential cryptanalysis, the boomerang attack, differential-linear cryptanalysis, and others).

1 Introduction

Most modern block ciphers are designed to resist a wide range of cryptanalytic techniques. Among them, one may cite linear cryptanalysis [19,20,23], differential cryptanalysis [7,8], as well as several variants such as impossible differentials [5], the boomerang attack [27] or the rectangle attack [6]. Proving resistance against all these attacks is often tedious and does not give any guarantee that a subtle new variant would not break the construction. Rather than considering all known attacks individually, it would obviously be preferable to give a *unique* proof, valid for a family of attacks.

In [26], Vaudenay shows that the decorrelation theory provides tools to prove security results in the Luby-Rackoff model [18], i.e., against adversaries only limited by the number of plaintext/ciphertext pairs they can access. Denoting d this number of pairs, the adversaries are referred to as *d -limited distinguishers*. Unfortunately, this class of adversaries does not capture the most widely studied statistical attacks such as linear and differential cryptanalysis. Instead, these attacks are formalized by so-called *iterated attacks of order d* [25]. This class of attacks was initially inspired by linear and differential cryptanalysis and actually formalizes most of the possible statistical attacks against block ciphers. For example, linear cryptanalysis is an iterated attack of order 1, differential cryptanalysis is of order 2, and higher order differential cryptanalysis [16,15] of order i is an iterated attack of order $d = 2^i$.

* Supported by the Swiss National Science Foundation, 200021-107982/1.

It is proven that resistance against all $2d$ -limited distinguishers is sufficient to resist iterated attacks of order d [26]. Consequently, designing a block cipher resistant to d -limited distinguishers for a large d is enough to resist most standard attacks against block ciphers. Obviously, this is not a trivial task as, to the best of our knowledge, no *efficient* block cipher was ever designed to resist d -limited distinguishers for $d > 2$ [14, 26].

In a previous article entitled “Dial C for Cipher” [1], we presented a block cipher construction provably resistant against (among others) linear and differential cryptanalysis (where the linear hull [21] and differentials [17] effects are taken into account, which is unfortunately not usual in typical proofs of security of block ciphers), several of their variants, 2-limited distinguishers and thus, all iterated attacks of order 1. Our aim in this article, is to design a block cipher based on the same principles as C but provably secure against d -limited distinguishers for large values of d . We call this construction KFC as it is based on a Feistel scheme. KFC is practical in the sense that it can be implemented and reach a throughput of a few Mbits/s. Just as the typical security proofs of block ciphers do not compare to ours, the encryption speed reached by KFC does not compare to those of nowadays block ciphers.

Constructions based on the decorrelation theory have already been proposed. COCONUT98 [24] was one of the first efficient block cipher based on decorrelation concepts. It resists 2-limited distinguishers but can be attacked by David Wagner’s boomerang attack [27], which is an iterated attack of order 4. Of course this does not prove that the decorrelation theory is useless, but only that decorrelation results do not prove more than what they claim. KFC is designed to resist d -limited distinguishers (and consequently, iterated attacks up to a given order), nothing more.

High Overview and Outline of the Paper. Before building a provably secure block cipher, we need to define precisely against which class of attacks it should be resistant. The adversary model and some reminders about the decorrelation theory are given in Section 2. Then, in Section 3, we give some hints about why we chose to use a Feistel scheme [13] for KFC. A description of the structure of the random functions we use in the Feistel scheme is then given in Section 4. The exact advantage of the best 2-limited distinguisher is computed in Section 5, and in Section 6, we bound the advantage of higher order adversaries.

2 Security Model

In this paper, a *perfectly random function* (resp. *permutation*) denotes a random function (resp. permutation) uniformly distributed among all possible functions (resp. permutations). Consequently, when referring to a *random function* or a *random permutation*, nothing is assumed about its distribution.

The Luby-Rackoff Model [18]. We consider an adversary \mathcal{A} with unbounded computational power, only limited by its number of queries d to an oracle \mathcal{O} implementing a random permutation. The goal of \mathcal{A} is to guess whether \mathcal{O} is implementing an instance drawn uniformly among the permutations defined by

a block cipher C or among all possible permutations, knowing that these two events have probability $\frac{1}{2}$ and that one of them is eventually true. Such an adversary is referred to as a d -limited adaptive distinguisher when he adaptively chooses his queries depending on previous answers from the Oracle or as a d -limited non-adaptive distinguisher when all the queries are made at once. In both cases, the ability of \mathcal{A} to succeed is measured by mean of its *advantage*.

Definition 1. *The advantage of \mathcal{A} of distinguishing two random functions F_0 and F_1 is defined by $\text{Adv}_{\mathcal{A}}(F_0, F_1) = |\Pr[\mathcal{A}(F_0) = 0] - \Pr[\mathcal{A}(F_1) = 0]|$.*

Informally, a secure block cipher C (i.e., a random permutation) should be indistinguishable from a perfectly random permutation C^* , i.e., the advantage $\text{Adv}_{\mathcal{A}}(C, C^*)$ of any adversary \mathcal{A} should be negligible. A secure random function F should be indistinguishable from a perfectly random function F^* , i.e., the advantage $\text{Adv}_{\mathcal{A}}(F, F^*)$ of any adversary \mathcal{A} should be negligible. Apart from very specific (and usually non-practical) constructions, computing the exact advantage of the best d -limited distinguisher is not straightforward. The decorrelation theory [26] gives some tools that will allow us to compute (or at least bound) this advantage for KFC.

Reminders on the Decorrelation Theory. Let $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a random function. The *distribution matrix* $[F]^d$ of F at order d is a $2^{nd} \times 2^{nd}$ matrix defined by $[F]^d_{(x_1, \dots, x_d), (y_1, \dots, y_d)} = \Pr_F[F(x_1) = y_1, \dots, F(x_d) = y_d]$. If F_1 and F_2 are two independent random functions, we have $[F_2 \circ F_1]^d = [F_1]^d \times [F_2]^d$. The advantage of the best distinguisher between F and F^* only depends on the *distance* between $[F]^d$ and $[F^*]^d$, whose exact definition will depend on whether the considered distinguisher is adaptive or not.

Definition 2. *Let $A \in \{0, 1\}^{nd} \times \{0, 1\}^{nd}$ be a matrix indexed by d -tuples of elements in $\{0, 1\}^n$. We let:*

$$\begin{aligned} \|A\|_{\infty} &= \max_{x_1, \dots, x_d} \sum_{y_1, \dots, y_d} |A_{(x_1, \dots, x_d), (y_1, \dots, y_d)}| \quad \text{and} \\ \|A\|_a &= \max_{x_1} \sum_{y_1} \dots \max_{x_d} \sum_{y_d} |A_{(x_1, \dots, x_d), (y_1, \dots, y_d)}|. \end{aligned}$$

Property 3 (Theorems 10 and 11 in [26]). *Let F be a random function and F^* be a perfectly random function. The advantage of the best d -limited non-adaptive distinguisher \mathcal{A} is such that $\text{Adv}_{\mathcal{A}}(F, F^*) = \frac{1}{2} \| [F]^d - [F^*]^d \|_{\infty}$ whereas the advantage of the best d -limited adaptive distinguisher \mathcal{A}_a is such that $\text{Adv}_{\mathcal{A}_a} = \frac{1}{2} \| [F]^d - [F^*]^d \|_a$.*

An iterated attack of order d consists in iterating independent non-adaptive d -limited attacks with random inputs. The algorithm of Fig. 1 gives a more formal definition of this concept. For example, linear cryptanalysis is an iterated attack of order 1 where $\mathcal{T}(X, Y) = a \cdot X \oplus b \cdot Y$ (where a and b respectively denote the input and output masks) and where X is a uniformly distributed random variable on text space. Similarly, differential cryptanalysis is an iterated attack

Parameters: a complexity n , a distribution on X , a test function \mathcal{T} outputting one bit, a set \mathcal{S}

Oracle: a permutation C

- 1: **for** $i = 1, \dots, n$ **do**
- 2: pick $X = (X_1, \dots, X_d)$ at random
- 3: get $Y = (C(X_1), \dots, C(X_d))$
- 4: set $T_i = \mathcal{T}(X, Y)$
- 5: **end for**
- 6: **if** $(T_1, \dots, T_n) \in \mathcal{S}$ **then** output 1 **else** output 0 **end if**

Fig. 1. Iterated attack of order d

of order 2 where $\mathcal{T}((X_1, X_2), (Y_1, Y_2))$ is 1 when $Y_1 \oplus Y_2 = b$ and 0 otherwise and where X_1 is a uniformly distributed random variable and $X_2 = X_1 \oplus a$. As proved in Theorem 18 in [26] bounding the advantage of the best $2d$ -limited non-adaptive adversary is sufficient to bound the advantage of any adversary performing an iterated attack of order d . Roughly speaking, a block cipher C with a negligible order $2d$ decorrelation $||| [C]^{2d} - [C^*]^{2d} |||_\infty$ is resistant to iterated attacks of order d .

3 From the SPN of C to the Feistel Scheme of KFC

The block cipher C (introduced in [1, 2]) is based on the same substitution-permutation network (SPN) as the AES [11], except that the fixed substitution boxes are replaced by mutually independent and perfectly random permutations. It achieves goals similar to those we want to achieve with KFC: being resistant against 2-limited adversaries, it is secure against all iterated attacks of order 1. These results were obtained by exploiting strong symmetries (induced by intrinsic symmetries of the confusion and diffusion layers) in the order 2 distribution matrix of C. Unfortunately, we were not able to exhibit similar symmetries for higher orders. It appears that layers of perfectly random permutations are suitable for proving security results at order 2, not above.

Instead of explicitly computing the advantage of a d -limited distinguisher we will try to bound it by a function of the advantage of the best $(d - 1)$ -limited distinguisher, and apply this bound recursively down to order 2 (which we know how to compute). This seems clearly impossible with layers of random permutations as two distinct inputs will always lead to two correlated outputs. However, this is not the case anymore when considering a layer of mutually independent and perfectly random *functions*. For instance, two distinct inputs of a perfectly random function yield two independent outputs. Similarly, if the two inputs of a layer of functions are distinct on each function input, the outputs are independent. This extends well to a set of d texts: if one text is different from *all* the others on *all* function inputs, the corresponding output is independent from all other outputs. A formal treatment of this idea is given in Section 4.

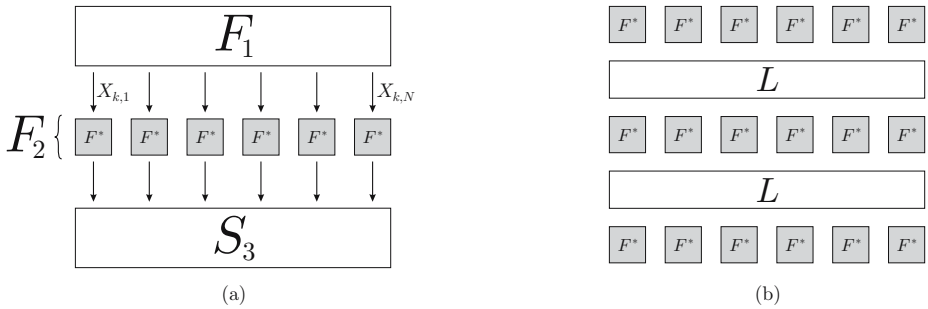


Fig. 2. Increasing the decorrelation order using a layer made of small *independent* and *perfectly random* functions

However, layers of random functions cannot always be inverted and thus do not fit in a classical SPN structure. The straightforward solution is to use a Feistel scheme [13]. Moreover, decorrelation results on the round functions of a Feistel scheme extend well to the whole construction.

Theorem 4 (Theorem 21 in [26]). *Let F^* be a uniformly distributed random function on $\{0, 1\}^n$. Let F_1, \dots, F_r be r independent random functions on $\{0, 1\}^n$ such that $\text{Adv}_{\mathcal{A}}(F_i, F^*) \leq \epsilon$ ($i = 1, \dots, r$) for any adversary \mathcal{A} . Let $C = \Psi(F_1, \dots, F_r)$ be an r round Feistel cipher on $\{0, 1\}^{2n}$. For any adversary \mathcal{A} limited to d queries and for any integer $k \geq 3$, we have:*

$$\text{Adv}_{\mathcal{A}}(C, C^*) \leq \frac{1}{2} \left(2k\epsilon + \frac{2d^2}{2^{n/2}} \right)^{\lfloor r/k \rfloor}.$$

This theorem shows that if we can instantiate *independent* random functions secure against all d -limited distinguishers, we can obtain a block cipher provably secure against any d -limited distinguisher. In the following sections, we focus on building a round function F_{KFC} following the ideas we have introduced here.

4 A Good Round Function F_{KFC} for the Feistel Scheme

To analyze the behavior of a layer of random functions, we analyze the construction $F = S_3 \circ F_2 \circ F_1$ where $F_1 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a random function, S_3 is a random permutation, and F_2 is a layer made of small independent and perfectly random functions (see Fig. 2(a)). We assume that F_1 , F_2 , and S_3 are mutually independent. We obtain an interesting property, making it possible to relate the order d decorrelation of F to its order $d - 1$ decorrelation. We consider a set of d inputs of the function F and denote the corresponding random outputs of F_1 by X_1, \dots, X_d , where $X_k = (X_{k,1}, \dots, X_{k,N})$ for $k = 1, \dots, d$. Let α be the event $\{\exists k \text{ s.t. } \forall j X_{k,j} \notin \{X_{1,j}, \dots, X_{k-1,j}, X_{k+1,j}, \dots, X_{d,j}\}\}$, that is, α is the event that one of the inputs is different from all the others on the N blocks. If α occurs, at least one of the outputs of the functions layer is a uniformly distributed

random variable independent from the others. More formally, if we denote \mathcal{A}_d the best d -limited adversary trying to distinguish F from F^* , we have:

$$\begin{aligned} \text{Adv}_{\mathcal{A}_d}(F, F^*) &= |1 - 2 \cdot \Pr[\mathcal{A}_d(F) = 1]| \\ &= |1 - 2 \cdot (\Pr[\mathcal{A}_d(F) = 1|\alpha] \Pr[\alpha] + \Pr[\mathcal{A}_d(F) = 1|\bar{\alpha}] \Pr[\bar{\alpha}])| \\ &\leq \text{Adv}_{\mathcal{A}_{d-1}}(F, F^*) \Pr[\alpha] + |1 - 2 \cdot \Pr[\mathcal{A}_d(F) = 1|\bar{\alpha}]| \Pr[\bar{\alpha}] \\ &\leq \text{Adv}_{\mathcal{A}_{d-1}}(F, F^*) + \Pr[\bar{\alpha}], \end{aligned} \tag{1}$$

where the first inequality comes from the fact that if α occurs, at least one output of F is completely independent from all the others. As S_3 is a permutation, it preserves this independence. Therefore, when α occurs, a d -limited distinguisher cannot be more efficient than the best $(d-1)$ -limited distinguisher (this is formally proven in Appendix A by looking at the definition of the decorrelation norms).

Why this is not Enough. From the previous inequality, it seems natural to consider a substitution-permutation-like construction made of an alternance of layers of independent and perfectly random functions and layers of linear diffusion (as shown on Fig. 2(b)). Intuitively, one could think that (as it is the case when iterating random permutations) iterating random functions is sufficient to decrease the advantage of a distinguisher. However, this is definitely *not* the case. Indeed, consider a 2-limited attack where the two plaintexts are equal on $N - 1$ blocks and different on the last block. There is a non-negligible probability $2^{-\ell}$ that, after the first layer of functions, both outputs are completely equal, thus leading to a distinguisher with advantage $2^{-\ell}$. For practical values of ℓ (e.g., $\ell = 8$), this is not acceptable. This means that we need a good resistance against 2-limited adversaries to initialize the recurrence relation of equation (1).

The Sandwich Technique. As proven in [1], an SPN using layers of mutually independent and perfectly random permutations is efficient against 2-limited distinguishers. Intuitively, this means that any set of d inputs will lead to a set of d *pairwise independent* outputs. As we will see in Section 6, pairwise independence is exactly what we need to apply the recursive relation (1).

For these reasons the construction we chose for F_{KFC} consists in sandwiching the construction sketched on Figure 2(b) between two SPN using layers of mutually independent and perfectly random permutations.

Description of F_{KFC} . The round function F_{KFC} used in the Feistel scheme defining KFC is based on three different layers:

- a substitution layer S made of N mutually independent and perfectly random ℓ bit permutations,
- a function layer F made of N mutually independent and perfectly random ℓ bit functions,
- a linear layer L which is a $N \times N$ matrix of elements in $\text{GF}(2^\ell)$ defining an MDS code (for optimal diffusion), which requires $N \leq 2^{\ell-1}$.

Let r_1 and r_2 be two integers. The round function F_{KFC} of KFC is defined as:

$$F_{\text{KFC}} = F_{\text{KFC}[r_1, r_2]} = S \circ (L \circ F)^{r_2} \circ (L \circ S)^{r_1}.$$

5 Computing the Advantage of the Best 2-Limited Distinguisher Against F_{KFC}

As all layers of F_{KFC} are mutually independent, the order 2 distribution matrix $[F_{\text{KFC}}]^2$ can be expressed as

$$[F_{\text{KFC}}]^2 = [S \circ (L \circ F)^{r_2} \circ (L \circ S)^{r_1}]^2 = ([S]^2 \times [L]^2)^{r_1} \times ([F]^2 \times [L]^2)^{r_2} \times [S]^2.$$

Each of these matrices is a $2^{2n} \times 2^{2n}$ square matrix, which makes direct computations impossible for practical parameters. In the rest of this Section we will exploit symmetries in order to reduce the computation to a product of $(N + 1) \times (N + 1)$ square matrices. For simplicity, we respectively denote by S , F , and L the distribution matrices $[S]^2$, $[F]^2$, and $[L]^2$ and let $q = 2^\ell$.

5.1 Conversion Matrices

Definition 5. *Considering $a \in \{0, 1\}^n$ as a N -tuple of elements in $\{0, 1\}^\ell$, the support of a is the binary N -tuple with 1's at the non-zero positions of a and 0 elsewhere. It is denoted $\text{SUPP}(a)$. The weight of the support, denoted $w(\text{SUPP}(a))$ or $w(a)$, is the Hamming weight of the support. When considering a pair $x, x' \in \{0, 1\}^n$, the support of the pair is $\text{SUPP}(x \oplus x')$.*

Distribution matrices at order 2 are indexed by pairs of texts. Using symmetries at two levels, we will first shrink them to $2^N \times 2^N$ matrices indexed by supports of pairs and then to $(N + 1) \times (N + 1)$ matrices indexed by weights. To do so, we define the following conversion matrices.

Pair of texts \leftrightarrow Support of pair. We let PS (resp. SP) denote the matrix that converts a pair of texts into a support (resp. a support into a pair of texts) in a uniform way. That is:

$$PS_{(x,x'),\gamma} = \mathbf{1}_{\gamma=\text{SUPP}(x\oplus x')} \quad \text{and} \quad SP_{\gamma',(y,y')} = \mathbf{1}_{\gamma'=\text{SUPP}(y\oplus y')} q^{-N} (q - 1)^{-w(\gamma')},$$

where $x, x', y, y' \in \{0, 1\}^n$ and $\gamma, \gamma' \in \{0, 1\}^N$. One can note that $SP \times PS = Id$.

Support of pair \leftrightarrow Weight. Similarly, we let WS (resp. SW) denote the matrix that converts a support into a weight (resp. a weight into a support) in a uniform way. That is:

$$SW_{\gamma,w} = \mathbf{1}_{w(\gamma)=w} \quad \text{and} \quad WS_{w',\gamma'} = \mathbf{1}_{w(\gamma')=w'} \binom{N}{w'}^{-1},$$

where $\gamma, \gamma' \in \{0, 1\}^N$ and $w, w' \in \{0, \dots, N\}$. We have $WS \times SW = Id$.

Pair of texts \leftrightarrow Weight. Finally we let $PW = PS \times SW$ and $WP = WS \times SP$ so that we obtain:

$$PW_{(x,x'),w} = \mathbf{1}_{w(x\oplus x')=w} \quad \text{and} \quad WP_{w',(y,y')} = \mathbf{1}_{w(y\oplus y')=w'} \binom{N}{w'}^{-1} q^{-N} (q - 1)^{-w'}.$$

Again, we have $WP \times PW = Id$.

5.2 Shrinking F and S, the First Step

Let $x, x', y, y' \in \text{GF}(q)^N$. As the N random functions of the F layer are mutually independent, we can express the coefficients of the distribution matrix F as

$$F_{(x,x'),(y,y')} = q^{-q \cdot N} \prod_{i=1}^N \#\{f_i : \text{GF}(q) \rightarrow \text{GF}(q) : f_i(x_i) = y_i, f_i(x'_i) = y'_i\}.$$

In the case where $\text{SUPP}(y \oplus y') \not\subseteq \text{SUPP}(x \oplus x')$, we have $F_{(x,x'),(y,y')} = 0$. When $\text{SUPP}(y \oplus y') \subseteq \text{SUPP}(x \oplus x')$, the uniform distribution of the f_i 's leads to:

$$F_{(x,x'),(y,y')} = q^{-q \cdot N} q^{-w(x \oplus x') + q \cdot N - N} = q^{-w(x \oplus x') - N},$$

and we see that F only depends on support of pairs. Consequently,

$$\begin{aligned} F_{(x,x'),(y,y')} &= \mathbf{1}_{\text{SUPP}(y \oplus y') \subseteq \text{SUPP}(x \oplus x')} q^{-w(x \oplus x') - N} \\ &= \sum_{\gamma, \gamma'} \mathbf{1}_{\gamma = \text{SUPP}(x \oplus x')} \mathbf{1}_{\gamma' = \text{SUPP}(y \oplus y')} \mathbf{1}_{\gamma' \subseteq \gamma} q^{-w(\gamma) - N} \\ &= \sum_{\gamma, \gamma'} PS_{(x,x'),\gamma} \mathbf{1}_{\gamma' \subseteq \gamma} q^{-w(\gamma)} (q-1)^{w(\gamma')} SP_{\gamma',(y,y')}. \end{aligned}$$

Defining the $2^N \times 2^N$ matrix \bar{F} by $\bar{F}_{\gamma,\gamma'} = \mathbf{1}_{\gamma' \subseteq \gamma} q^{-w(\gamma)} (q-1)^{w(\gamma')}$ we obtain:

$$F = PS \times \bar{F} \times SP. \tag{2}$$

Similarly, for the S layer we have:

$$S_{(x,x'),(y,y')} = \mathbf{1}_{\text{SUPP}(x \oplus x') = \text{SUPP}(y \oplus y')} q^{-N} (q-1)^{-w(x \oplus x')} = \sum_{\gamma} PS_{(x,x'),\gamma} SP_{\gamma,(y,y')}$$

and thus,

$$S = PS \times SP. \tag{3}$$

5.3 Shrinking L, the Second Step

Given the structure of F_{KFC} , each linear layer L is surrounded by S or F layers. From equations (2) and (3), this means that each matrix L is surrounded by the conversion matrices PS and SP. Denoting $\bar{L} = SP \times L \times PS$ we obtain:

$$\begin{aligned} \bar{L}_{\gamma,\gamma'} &= \sum_{(x,x')} \sum_{(y,y')} SP_{\gamma,(x,x')} L_{(x,x'),(y,y')} PS_{(y,y'),\gamma'} \\ &= q^{-N} (q-1)^{-w(\gamma)} \sum_{(x,x')} \mathbf{1}_{\gamma = \text{SUPP}(x \oplus x')} \mathbf{1}_{\gamma' = \text{SUPP}(L(x \oplus x'))} \\ &= (q-1)^{-w(\gamma)} \sum_x \mathbf{1}_{\gamma = \text{SUPP}(x)} \mathbf{1}_{\gamma' = \text{SUPP}(L(x))}. \end{aligned}$$

The sum in this equation is the number of texts of a given support γ that are mapped by the MDS linear layer L on a text of support γ' . The number of codewords with given supports can be explicitly computed for any MDS code (see Theorem 3 in [12]) and, amazingly, only depends on the weights of the supports γ and γ' . We obtain the following formula:

$$\bar{L}_{\gamma,\gamma'} = (q - 1)^{-w(\gamma)} \frac{E(w(\gamma) + w(\gamma'))}{\binom{2N}{w(\gamma)+w(\gamma')}} ,$$

where $E(i) = \binom{2N}{i} \sum_{j=N+1}^i \binom{i}{j} (-1)^{i-j} (q^{j-N} - 1)$ for $i > N$, $E(0) = 1$, and $E(i) = 0$ for $0 < i \leq N$. As the previous equation only depends on the weights of γ and γ' , we can shrink L even more:

$$\begin{aligned} \bar{L}_{\gamma,\gamma'} &= \sum_{w,w'} \mathbf{1}_{w(\gamma)=w} \mathbf{1}_{w(\gamma')=w'} (q - 1)^{-w} \frac{E(w + w')}{\binom{2N}{w+w'}} \\ &= \sum_{w,w'} SW_{\gamma,w} \binom{N}{w'} (q - 1)^{-w} \frac{E(w + w')}{\binom{2N}{w+w'}} WS_{w',\gamma'} . \end{aligned}$$

Defining the $(N + 1) \times (N + 1)$ matrix \bar{L} by $\bar{L}_{w,w'} = \binom{N}{w'} (q - 1)^{-w} \frac{E(w+w')}{\binom{2N}{w+w'}}$,

$$\bar{L} = SW \times \bar{L} \times WS. \tag{4}$$

A Brief Summary of the Situation. We started from $[F_{\text{KFC}}]^2 = (S \times L)^{r_1} \times (F \times L)^{r_2} \times S$. To make things clearer, we consider the case where $r_1 = 1$ and $r_2 = 2$. Using equations (2), (3), and (4) we obtain:

$$\begin{aligned} [F_{\text{KFC}}]^2 &= S \times L \times F \times L \times F \times L \times S \\ &= PS \times SP \times L \times PS \times \bar{F} \times SP \times L \times PS \times \bar{F} \times SP \times L \times PS \times SP \\ &= PS \times SW \times \bar{L} \times WS \times \bar{F} \times SW \times \bar{L} \times WS \times \bar{F} \times SW \times \bar{L} \times WS \times SP \\ &= PW \times \bar{L} \times WS \times \bar{F} \times SW \times \bar{L} \times WS \times \bar{F} \times SW \times \bar{L} \times WP . \end{aligned}$$

Now we focus on the simplification of $WS \times \bar{F}$.

5.4 Shrinking $WS \times \bar{F}$, the Third (and Last) Step

We have:

$$\begin{aligned} (WS \times \bar{F})_{w,\gamma'} &= \sum_{\gamma} WS_{w,\gamma} \bar{F}_{\gamma,\gamma'} = \binom{N}{w}^{-1} q^{-w} (q - 1)^{w(\gamma')} \sum_{\gamma} \mathbf{1}_{w(\gamma)=w} \mathbf{1}_{\gamma' \subseteq \gamma} \\ &= \binom{N}{w}^{-1} q^{-w} (q - 1)^{w(\gamma')} \mathbf{1}_{w \geq w(\gamma')} \binom{N-w(\gamma')}{N-w} , \end{aligned}$$

so that $(WS \times \bar{F})_{w,\gamma'}$ only depends on w and on the *weight* of γ' . Consequently, letting $\bar{\bar{F}}$ be the $(N + 1) \times (N + 1)$ matrix defined by $\bar{\bar{F}}_{w,w'} = q^{-w} (q - 1)^{w'} \mathbf{1}_{w \geq w'} \binom{w}{w'}$, we obtain:

$$WS \times \bar{F} = \bar{\bar{F}} \times WS.$$

Final Summary of the Situation. From the previous summary and the last shrinking step, we finally obtain that:

$$\begin{aligned}
 [F_{\text{KFC}}]^2 &= PW \times \bar{L} \times \bar{F} \times WS \times SW \times \bar{L} \times \bar{F} \times WS \times SW \times \bar{L} \times WP \\
 &= PW \times \bar{L} \times \bar{F} \times \bar{L} \times \bar{F} \times \bar{L} \times WP.
 \end{aligned}$$

In the general case, this means that $[F_{\text{KFC}}]^2 = PW \times (\bar{L})^{r_1} \times (\bar{F} \times \bar{L})^{r_2} \times WP$.

5.5 Practical Computation of the Advantage

The expression we just obtained for $[F_{\text{KFC}}]^2$ leads to a simple practical expression for $\|[F_{\text{KFC}}]^2 - [F^*]^2\|_a$. Noting that an adversary cannot increase his advantage asking twice the same query, we have:

$$\|[F_{\text{KFC}}]^2 - [F^*]^2\|_a = \max_x \sum_y \max_{x' \neq x} \sum_{y'} \left| [F_{\text{KFC}}]_{(x,x'),(y,y')}^2 - q^{-2N} \right|.$$

Let U be the $(N + 1) \times (N + 1)$ matrix defined by $U_{w,w'} = q^{-N}(q - 1)w' \binom{N}{w}$, so that for all x, x', y, y' we have $(PW \times U \times WP)_{(x,x'),(y,y')} = q^{-2N}$. Consequently, $\|[F_{\text{KFC}}]^2 - [F^*]^2\|_a$ is equal to:

$$\max_x \sum_y \max_{x' \neq x} \sum_{y'} \left| \left(PW \times ((\bar{L})^{r_1} \times (\bar{F} \times \bar{L})^{r_2} - U) \times WP \right)_{(x,x'),(y,y')} \right|.$$

As the inner matrix only depends on $w(x \oplus x')$ and of $w(y \oplus y')$, we get

$$\|[F_{\text{KFC}}]^2 - [F^*]^2\|_a = \max_{w \neq 0} \sum_{w'} \left| \left((\bar{L})^{r_1} \times (\bar{F} \times \bar{L})^{r_2} - U \right)_{w,w'} \right|$$

Similar computations show that $\|[F_{\text{KFC}}]^2 - [F^*]^2\|_\infty = \|[F_{\text{KFC}}]^2 - [F^*]^2\|_a$.

Theorem 6. *Let \bar{L} , \bar{F} , and U be $(N + 1) \times (N + 1)$ matrices defined as above. The advantage of the best 2-limited distinguisher \mathcal{A} (whether adaptive or not) against $F_{\text{KFC}} = S \circ (L \circ F)^{r_2} \circ (L \circ S)^{r_1}$ is given by:*

$$\text{Adv}_{\mathcal{A}}(F_{\text{KFC}}, F^*) = \frac{1}{2} \max_{w \neq 0} \sum_{w'} \left| \left((\bar{L})^{r_1} \times (\bar{F} \times \bar{L})^{r_2} - U \right)_{w,w'} \right|.$$

Explicit values of this advantage for some typical values of N, q, r_1 and r_2 are given in Table 1. We note that $r_1 = 3$ is enough (at least for these parameters). Moreover, the advantage increases with the value of r_2 . The reason is that the more F layers there is, the higher is the probability of an internal collision.

6 Bounding the Advantage of the Best d -Limited Distinguisher Against F_{KFC} for $d > 2$

6.1 Replacing F by F o S

To simplify the proofs, we will replace each F layer of F_{KFC} by F o S. Both constructions are completely equivalent in the sense that any decorrelation result

Table 1. Advantage of the best 2-limited distinguisher against F_{KFC}

		$N = 8$ and $q = 2^8$				$N = 8$ and $q = 2^{16}$				$N = 16$ and $q = 2^8$			
		0	1	10	100	0	1	10	100	0	1	10	100
$r_2 \backslash r_1$	0	1	2^{-5}	2^{-8}	2^{-8}	1	2^{-13}	2^{-16}	2^{-16}	1	2^{-4}	2^{-8}	2^{-8}
	1	2^{-5}	2^{-50}	2^{-52}	2^{-49}	2^{-13}	2^{-114}	2^{-116}	2^{-113}	2^{-4}	2^{-95}	2^{-104}	2^{-103}
	2	2^{-46}	2^{-53}	2^{-52}	2^{-49}	2^{-110}	2^{-117}	2^{-116}	2^{-113}	2^{-87}	2^{-104}	2^{-104}	2^{-103}
	3	2^{-62}	2^{-53}	2^{-52}	2^{-49}	2^{-128}	2^{-117}	2^{-116}	2^{-113}	2^{-120}	2^{-104}	2^{-104}	2^{-103}

holding for the latter also holds for the original construction, the reason being that $[F \circ S]^d = [F]^d$ (see Appendix B for a proof). From now on, we thus study the following equivalent construction:

$$F_{KFC} = F_{KFC[r_1, r_2]} = S \circ (L \circ F \circ S)^{r_2} \circ (L \circ S)^{r_1}.$$

Assumption 7. For $r_1 > 2$, any $i \in \{0, \dots, r_2\}$ and any 2-limited distinguisher \mathcal{A}_2 , we have $\text{Adv}_{\mathcal{A}_2}(F_{KFC[r_1, r_2]}, F^*) \geq \text{Adv}_{\mathcal{A}_2}(F_{KFC[r_1, i]}, F^*)$.

This assumption seems natural from Table 1, although it might prove wrong in the general case (in particular, the threshold for r_1 might be different for other values of N and q). However, we experimentally verified it for all values of the parameters we consider in the rest of this paper.

In practice, Assumption 7 means that, when the advantage of the best 2-limited distinguisher against F_{KFC} is negligible, this is also the case before any F layer. The inputs of any F layer can thus be considered as *pairwise independent*.

6.2 Taking Advantage of the Pairwise Independence

Let $i \in \{0, \dots, r_2\}$. Referring to Section 4, we denote α_{i-1} the event α and let $F_1 = F_{KFC[r_1, i-1]}$, $F_2 = F$, and $S_3 = S \circ L$. We use these notations, $F_{KFC[r_1, i]} = S_3 \circ F_2 \circ F_1$, so that equation (1) gives

$$\text{Adv}_{\mathcal{A}_d}(F_{KFC[r_1, i]}, F^*) \leq \text{Adv}_{\mathcal{A}_{d-1}}(F_{KFC[r_1, i]}, F^*) + \Pr[\bar{\alpha}_{i-1}].$$

Bounding $\Pr[\bar{\alpha}_{i-1}]$ for all i allows to recursively bound $\text{Adv}_{\mathcal{A}_d}(F_{KFC[r_1, i]}, F^*)$. As in Section 4, we denote the output of F_1 by X_1, \dots, X_d where, for $k = 1, \dots, d$, we have $X_k = (X_{k,1}, \dots, X_{k,N})$. Let $0 \leq \lambda \leq d$ be the number of X_k 's different from all other texts on all N blocks. We have:

$$\lambda = \sum_{k=1}^d \prod_{b=1}^N \prod_{\substack{j=1 \\ j \neq k}}^d \mathbf{1}_{X_{k,b} \neq X_{j,b}}.$$

Using the linearity of the mean and the mutual independence of the N blocks, we obtain $E(\lambda) = d \cdot (\Pr[X_{1,1} \notin \{X_{2,1}, \dots, X_{d,1}\}])^N$.

Property 8. For $d > 0$ we have $\mathcal{P}_d = \Pr[X_{1,1} \notin \{X_{2,1}, \dots, X_{d,1}\}] \geq 1 - \frac{d-1}{q}$ and thus, $E(\lambda) \geq d \cdot \left(1 - \frac{d-1}{q}\right)^N$.

Proof. The proof is done by induction on d . For $d = 1$ the result is trivial. Assume $\mathcal{P}_d \geq 1 - (d - 1)/q$ for an arbitrary d . As stated in Section 6.1, we can assume that the X_i 's are pairwise independent and thus:

$$\begin{aligned} \mathcal{P}_{d+1} &= \mathcal{P}_d - \Pr[X_{1,1} \notin \{X_{2,1}, \dots, X_{d,1}\}, X_{1,1} = X_{d+1,1}] \\ &\geq \mathcal{P}_d - \Pr[X_{1,1} = X_{d+1,1}] = \mathcal{P}_d - \frac{1}{q}. \end{aligned}$$

The expression we obtained for $E(\lambda)$ leads to the final result. □

Using this result, we can easily bound $\Pr[\bar{\alpha}_i]$ as $E(\lambda) = \sum_{k=1}^d k \Pr[\lambda = k] \leq d \Pr[\lambda \neq 0] = d \Pr[\alpha_i]$, so that, for all $i \in \{0, \dots, r_2\}$,

$$\Pr[\bar{\alpha}_i] \leq 1 - \frac{E(\lambda)}{d} \leq 1 - \left(1 - \frac{d-1}{q}\right)^N. \tag{5}$$

6.3 Piling-Up the Rounds

Obviously, the bound on $\Pr[\bar{\alpha}_i]$ we just obtained cannot be used directly to obtain a meaningful bound on the advantage of high order distinguishers. Consequently, we will consider t successive α_i events and give an upper bound on the probability that *none* of them occurs. We have $\Pr[\bar{\alpha}_1, \dots, \bar{\alpha}_t] = \Pr[\bar{\alpha}_t | \bar{\alpha}_1, \dots, \bar{\alpha}_{t-1}] \cdot \Pr[\bar{\alpha}_1, \dots, \bar{\alpha}_{t-1}]$. As the bound on $E(\lambda)$ only relies on the pairwise independence of the inputs of the i -th round, the bound given by equation (5) can also be proven for $\Pr[\bar{\alpha}_t | \bar{\alpha}_1, \dots, \bar{\alpha}_{t-1}]$. By induction, we finally obtain that:

$$\Pr[\bar{\alpha}_1, \dots, \bar{\alpha}_t] \leq \left(1 - \left(1 - \frac{d-1}{q}\right)^N\right)^t.$$

Theorem 9. Assume that the advantage of the best 2-limited distinguisher on $F_{\text{KFC}[r_1, r_2]}$ is bounded by ϵ . For any d and set of integers $\{t_3, \dots, t_d\}$ such that $\sum_{i=3}^d t_i \leq r_2$, the advantage of the best d -limited distinguisher \mathcal{A}_d on $F_{\text{KFC}[r_1, r_2]}$ is such that:

$$\text{Adv}_{\mathcal{A}_d}(F_{\text{KFC}[r_1, r_2]}, F^*) \leq \epsilon + \sum_{i=3}^d \left(1 - \left(1 - \frac{i-1}{q}\right)^N\right)^{t_i}.$$

Fixing $r_1 = 3$, the previous theorem bounds, for any value of d , the advantage of the best d -limited distinguisher against a given number of rounds r_2 of F_{KFC} . In Table 2 we give the best bounds we obtain for various values of r_2 , d , N , and q . If one aims at a specific value of d and wants to select r_2 in order to bound the advantage of the best d -limited distinguisher, the best choice is probably to select the t_i 's such that $\Pr[\bar{\alpha}_1, \dots, \bar{\alpha}_{t_i}] < \epsilon$, which bounds the advantage by $d \cdot \epsilon$. The following theorem generalizes this idea.

Table 2. Bounds on $\text{Adv}_{\mathcal{A}_d}$ for $r_1 = 3$ and various parameters

		$N = 8$ and $q = 2^8$						$N = 8$ and $q = 2^{16}$						
$r_2 \backslash d$	2	3	4	8	16	32	64	2	3	4	8	16	32	64
10	2^{-52}	2^{-40}	2^{-17}	2^{-2}	1	1	1	2^{-116}	2^{-116}	2^{-57}	2^{-11}	1	1	1
100	2^{-49}	2^{-49}	2^{-49}	2^{-46}	2^{-11}	1	1	2^{-113}	2^{-113}	2^{-113}	2^{-113}	2^{-66}	2^{-23}	2^{-5}
250	2^{-48}	2^{-48}	2^{-48}	2^{-48}	2^{-33}	2^{-5}	1	2^{-112}	2^{-112}	2^{-112}	2^{-112}	2^{-112}	2^{-69}	2^{-25}
1000	2^{-46}	2^{-46}	2^{-46}	2^{-46}	2^{-46}	2^{-35}	2^{-2}	2^{-110}	2^{-110}	2^{-110}	2^{-110}	2^{-110}	2^{-110}	2^{-110}

		$N = 16$ and $q = 2^8$					
$r_2 \backslash d$	2	3	4	8	16	32	64
10	2^{-104}	2^{-31}	2^{-12}	1	1	1	1
100	2^{-103}	2^{-103}	2^{-103}	2^{-31}	2^{-5}	1	1
250	2^{-103}	2^{-103}	2^{-103}	2^{-81}	2^{-18}	1	1
1000	2^{-102}	2^{-102}	2^{-102}	2^{-102}	2^{-82}	2^{-12}	1

Theorem 10. Assume that the advantage of the best 2-limited distinguisher against $F_{\text{KFC}[r_1, r_2]}$ is bounded by ϵ . Let:

$$t_d(\beta) = \min_t \{ \Pr[\bar{\alpha}_1, \dots, \bar{\alpha}_t] < \beta \cdot \epsilon \} = \left\lceil \frac{\log(\beta \cdot \epsilon)}{\log\left(1 - \left(1 - \frac{d-1}{q}\right)^N\right)} \right\rceil.$$

For any d such that $\sum_{i=3}^d t_i(\beta) \leq r_2$, the advantage of the best d -limited distinguisher \mathcal{A}_d against $F_{\text{KFC}[r_1, r_2]}$ is such that:

$$\text{Adv}_{\mathcal{A}_d}(F_{\text{KFC}[r_1, r_2]}, F^*) \leq \epsilon + \sum_{i=3}^d \left(1 - \left(1 - \frac{i-1}{q}\right)^N\right)^{t_i(\beta)} \leq \epsilon \cdot (1 + (d-2) \cdot \beta).$$

7 Conclusion

We introduced KFC, a block cipher based on a three round Feistel scheme. Each of the three round functions has an SPN-like structure for which we can either compute or bound the advantage of the best d -limited adaptive adversary, for any value of d . Using results from the Decorrelation Theory, we extend these results to the whole KFC construction. At this time, no key schedule has been specified for KFC. We suggest to use the same trick as in [1], i.e., use a key schedule based on a cryptographically secure pseudo-random generator (for example the good old BBS [10] or a faster generator like QUAD [4, 3]). This way, all the results we have proven assuming the mutual independence of the random functions and permutations remain valid when implementing KFC in practice with a 128 bit secret key. We propose two sets of parameters:

Regular KFC: $N = 8$, $q = 2^8$, $r_1 = 3$, $r_2 = 100$. These parameters lead to provable security against 8-limited adaptive distinguishers. Consequently,

Regular KFC is resistant to iterated attacks of order 4, which include linear and differential cryptanalysis, the boomerang attack and others. Based on existing implementation results on C, we estimate the encryption speed of Regular KFC to 15-25 Mbits/s on a Pentium IV 2GHz. The key schedule needs to generate approximately 2^{22} cryptographically secure pseudo-random bits.

Extra Crispy KFC: $N = 8$, $q = 2^{16}$, $r_1 = 3$, $r_2 = 1000$. Using these quite *extreme* parameters, we manage to obtain provable security against 70-limited adaptive adversaries, but encryption rate could probably never reach more than 1 Mbit/s. Also, the key schedule should produce 2^{35} pseudo random bits, which means that Extra Crispy KFC requires at least 4 GB of memory.

To the best of our knowledge, KFC is the first practical block cipher to propose tight security proofs of resistance against large classes of attacks, including most classical cryptanalysis (such as linear and differential cryptanalysis, taking hull effect in consideration in both cases, higher order differential cryptanalysis, the boomerang attack, differential-linear cryptanalysis, or the rectangle attack).

References

1. T. Baignères and M. Finiasz. Dial C for Cipher. In Biham and Youssef [9]. To appear.
2. T. Baignères and S. Vaudenay. Proving the security of AES substitution-permutation network. In B. Preneel and S.E. Tavares, editors, *Selected Areas in Cryptography, SAC 05*, volume 3897 of *LNCS*, pages 65–81. Springer-Verlag, 2006.
3. C. Berbain, O. Billet, and H. Gilbert. Efficient implementations of multivariate quadratic systems. In Biham and Youssef [9]. To appear.
4. C. Berbain, H. Gilbert, and J. Patarin. QUAD: a practical stream cipher with provable security. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT '06*, volume 4004 of *LNCS*, pages 109–128. Springer-Verlag, 2006.
5. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In Stern [22], pages 12–23.
6. E. Biham, O. Dunkelman, and N. Keller. The rectangle attack - rectangling the Serpent. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT '01*, volume 2045 of *LNCS*, pages 340–357. Springer-Verlag, 2001.
7. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4:3–72, 1991.
8. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems (extended abstract). In A. Menezes and S. Vanstone, editors, *Advances in Cryptology - CRYPTO '90*, volume 537 of *LNCS*, pages 2–21. Springer-Verlag, 1991.
9. E. Biham and A.M. Youssef, editors. *Proceedings of Selected Areas in Cryptography, SAC 06*, LNCS. Springer-Verlag, 2006. To appear.
10. L. Blum, M. Blum, and M. Shub. Comparison of two pseudo-random number generators. In D. Chaum, R.L. Rivest, and A. Sherman, editors, *Advances in Cryptology - CRYPTO '82*, pages 61–78. Plemum, 1983.
11. J. Daemen and V. Rijmen. *The Design of Rijndael*. Information Security and Cryptography. Springer-Verlag, 2002.
12. M. El-Khamy and R. McEliece. The partition weight enumerator of MDS codes and its applications. In *IEEE International Symposium on Information Theory, ISIT 2005*. IEEE, 2005. Available on <http://arxiv.org/pdf/cs.IT/0505054>.

13. H. Feistel. Cryptography and computer privacy. *Scientific American*, 228:15–23, 1973.
14. L. Granboulan, P. Nguyen, F. Noilhan, and S. Vaudenay. DFCv2. In D.R. Stinson and S.E. Tavares, editors, *Selected Areas in Cryptography, SAC'00*, volume 2012 of *LNCS*, pages 57–71. Springer-Verlag, 2001.
15. L. Knudsen. Truncated and higher order differentials. In B. Preneel, editor, *Fast Software Encryption - FSE'95*, volume 1008 of *LNCS*, pages 196–211. Springer-Verlag, 1995.
16. X. Lai. Higher order derivatives and differential cryptanalysis. In Kluwer Academic Publishers, editor, *Symposium on Communication, Coding and Cryptography*, pages 227–233, 1994.
17. X. Lai, J. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In D.W. Davies, editor, *Advances in Cryptology - EUROCRYPT'91*, volume 547 of *LNCS*, pages 17–38. Springer-Verlag, 1991.
18. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.
19. M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y.G. Desmedt, editor, *Advances in Cryptology - CRYPTO'94*, volume 839 of *LNCS*, pages 1–11. Springer-Verlag, 1994.
20. M. Matsui. Linear cryptanalysis method for DES cipher. In T. Hellesest, editor, *Advances in Cryptology - EUROCRYPT'93*, volume 765 of *LNCS*, pages 386–397. Springer-Verlag, 1994.
21. K. Nyberg. Linear approximation of block ciphers. In A. De Santis, editor, *Advances in Cryptology - EUROCRYPT'94*, volume 950 of *LNCS*, pages 439–444. Springer-Verlag, 1995.
22. J. Stern, editor. *Proceedings of Advances in Cryptology - EUROCRYPT'99*, volume 1592 of *LNCS*. Springer-Verlag, 1999.
23. A. Tardy-Corffdir and H. Gilbert. A known plaintext attack of FEAL-4 and FEAL-6. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO'91*, volume 576 of *LNCS*, pages 172–182. Springer-Verlag, 1992.
24. S. Vaudenay. Provable security for block ciphers by decorrelation. In *STACS'98*, volume 1373 of *LNCS*, pages 249–275. Springer-Verlag, 1998.
25. S. Vaudenay. Resistance against general iterated attacks. In Stern [22], pages 255–271.
26. S. Vaudenay. Decorrelation: a theory for block cipher security. *Journal of Cryptology*, 16(4):249–286, 2003.
27. D. Wagner. The boomerang attack. In L. Knudsen, editor, *Fast Software Encryption - FSE'99*, volume 1636 of *LNCS*, pages 156–170. Springer-Verlag, 1999.

A Proof of $|1 - 2 \cdot \Pr[\mathcal{A}_d(F) = 1 \mid \alpha]| = \text{Adv}_{\mathcal{A}_{d-1}}(F, F^*)$

Without loss of generality, we can assume that the adversary does not make the same query twice (as this would not increase its advantage) and that the event α is true for the d th query. In this case, we know that $(F_2 \circ F_1)(x_d)$ is a uniformly distributed random variable independent of $(F_2 \circ F_1)(x_i)$ for all $i < d$. As S_3 is a permutation, this property is still true for $(S_3 \circ F_2 \circ F_1)(x_d) = F(x_d)$. Denoting by Y this random variable we have:

$$\begin{aligned} \Pr[F(x_1) = y_1, \dots, F(x_d) = y_d \mid \alpha] &= \Pr[F(x_1) = y_1 \dots F(x_{d-1}) = y_{d-1}, Y = y_d] \\ &= 2^{-n} \Pr[F(x_1) = y_1 \dots F(x_{d-1}) = y_{d-1}]. \end{aligned}$$

Let $A = |1 - 2 \cdot \Pr[\mathcal{A}_d(F) = 1 \mid \alpha]|$. Similarly to the proof of Theorem 10 in [26] we know that:

$$A = \frac{1}{2} \max_{x_1} \sum_{y_1} \cdots \max_{x_d} \sum_{y_d} |\Pr[F(x_1) = y_1, \dots, F(x_d) = y_d \mid \alpha] - 2^{-d \cdot n}|.$$

From the two previous equations we obtain that:

$$\begin{aligned} A &= \frac{1}{2} \max_{x_1} \sum_{y_1} \cdots \max_{x_d} \sum_{y_d} 2^{-n} \left| \Pr[F(x_1) = y_1 \dots F(x_{d-1}) = y_{d-1}] - 2^{-(d-1) \cdot n} \right| \\ &= \frac{1}{2} \max_{x_1} \sum_{y_1} \cdots \max_{x_{d-1}} \sum_{y_{d-1}} \left| \Pr[F(x_1) = y_1 \dots F(x_{d-1}) = y_{d-1}] - 2^{-(d-1) \cdot n} \right| \\ &= \text{Adv}_{\mathcal{A}_{d-1}}(F, F^*). \end{aligned}$$

B Proof That $[\text{F} \circ \text{S}]^d = [\text{F}]^d$

For any $x = (x_1, \dots, x_d), y = (y_1, \dots, y_d) \in \{0, 1\}^{nd}$ we have:

$$\begin{aligned} [\text{F} \circ \text{S}]^d_{(x,y)} &= \Pr[(x_1, \dots, x_d) \xrightarrow{\text{F} \circ \text{S}} (y_1, \dots, y_d)] \\ &= \prod_{i=1}^d \Pr[(x_{1,i}, \dots, x_{d,i}) \xrightarrow{F^* \circ C^*} (y_{1,i}, \dots, y_{d,i})] \\ &= \prod_{i=1}^d \frac{1}{2^{\ell!}} \sum_c \Pr[(c(x_{1,i}), \dots, c(x_{d,i})) \xrightarrow{F^*} (y_{1,i}, \dots, y_{d,i})] \\ &= \prod_{i=1}^d \frac{1}{2^{\ell!}} \sum_c \Pr[(x_{1,i}, \dots, x_{d,i}) \xrightarrow{F^*} (c^{-1}(y_{1,i}), \dots, c^{-1}(y_{d,i}))] \\ &= \prod_{i=1}^d \Pr[(x_{1,i}, \dots, x_{d,i}) \xrightarrow{F^*} (y_{1,i}, \dots, y_{d,i})] \\ &= \Pr[(x_1, \dots, x_d) \xrightarrow{\text{F}} (y_1, \dots, y_d)] \\ &= [\text{F}]^d_{(x,y)} \end{aligned}$$

Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions

Jacques Patarin¹, Valérie Nacheff², and Côme Berbain³

¹ Université de Versailles

45 avenue des Etats-Unis, 78035 Versailles Cedex, France

² Université de Cergy-Pontoise

2 avenue Adolphe Chauvin, 95011 Cergy-Pontoise Cedex, France

³ France Telecom Research and Development

38-40 rue du Général Leclerc, 92794 Issy-les-Moulineaux, France

jacques.patarin@prism.uvsq.fr, valerie.nacheff@u-cergy.fr,
come.berbain@orange-ft.com

Abstract. In this paper, we describe generic attacks on unbalanced Feistel schemes with contracting functions. These schemes are used to construct pseudo-random permutations from kn bits to kn bits by using d pseudo-random functions from $(k - 1)n$ bits to n bits. We describe known plaintext attacks (KPA) and non-adaptive chosen plaintext attacks (CPA-1) against these schemes with less than 2^{kn} plaintext/ciphertext pairs and complexity strictly less than $O(2^{kn})$ for a number of rounds $d \leq 2k - 1$. Consequently at least $2k$ rounds are necessary to avoid generic attacks. For $k = 3$, we found attacks up to 6 rounds, so 7 rounds are required. When $d \geq 2k$, we also describe some attacks on schemes with generators, (i.e. schemes where the d pseudo-random functions are generated) and where more than one permutation is required.

Keywords: unbalanced Feistel permutations, pseudo-random permutations, generic attacks, Luby-Rackoff theory, block ciphers.

1 Introduction

Feistel schemes are widely used in symmetric cryptography in order to construct pseudo-random permutations. In trying to design such scheme, one of the natural questions is: what is the the minimum number of rounds required to avoid all the “generic attacks”. By generic attacks we mean all the attacks effective with high probability when the round functions are randomly chosen. We are mainly interested in generic attacks with a complexity that is much smaller than a search on all possible inputs of the permutation.

Many results are known on classical (balanced) Feistel schemes. In [7], Luby and Rackoff have shown their famous result: for more than 3 rounds all the generic chosen plaintext attacks on Feistel schemes require at least $O(2^{\frac{n}{2}})$ inputs. Moreover for more than 4 rounds all the generic attacks on adaptive chosen plaintext/ciphertext require at least $O(2^{\frac{n}{2}})$ inputs. These bounds are tight [1,10].

It has also been proved that to avoid all attacks with less than 2^{2n} computations at least 6 rounds of balanced Feistel schemes are needed [2,11,12]. This result is still valid if the round functions are permutations [5,6]. For more than 6 rounds, some attacks are still possible but with more than 2^{2n} computations [11]. All these results on classical Feistel schemes are summarized in Table 1:

Table 1. Results (from [12]) on G_2^d . For more than 6 rounds more than one permutation is needed or more than 2^{2n} computations are needed in the best known attacks to distinguish G_2^d from a random permutation with an even signature.

	KPA	CPA-1	CPCA-2
G_2^1	1	1	1
G_2^2	$2^{\frac{n}{2}}$	2	2
G_2^3	$2^{\frac{n}{2}}$	$2^{\frac{n}{2}}$	3
G_2^4	2^n	$2^{n/2}$	$2^{\frac{n}{2}}$
G_2^5	$2^{3n/2}$	2^n	2^n
G_2^6	2^{2n}	2^{2n}	2^{2n}
G_2^7	2^{3n}	2^{3n}	2^{3n}
G_2^8	2^{4n}	2^{4n}	2^{4n}
$G_2^d, d \geq 8$	$2^{(k-4)n}$	$2^{(k-4)n}$	$2^{(k-4)n}$

The aim of this paper is to look for similar results for the case of unbalanced Feistel schemes with contracting functions: we call such schemes “contracting Feistel Schemes”. A precise definition of these schemes is given in Sect. 2. The case of unbalanced Feistel schemes with expanding functions instead of contracting functions is studied in [4,14,15]. Some results on contracting Feistel schemes or on small transformations of these schemes can be found in [8,9]. In [9], Naor and Reingold studied the security of contracting Feistel schemes with pairwise independent permutations. They show lower bounds for the security of such schemes. Lucks [8] gives some security results on contracting Feistel schemes built with hash functions.

The paper is organized as follows. In Sect. 2 and 3, we introduce notations and present precise definitions of the considered schemes and an overview of our attacks. In Sect. 4, we study attacks for $k = 3$ and $d \leq 6$. Then in Sect. 5, we give attacks for any k and $d \leq 2k - 1$. Finally, Sect. 6 is devoted to what can be done with more than 2^{kn} computations. In particular, we describe attacks against permutation generators. All the results are summarized in the conclusion: these tables extend the above Table 1 to the case of unbalanced Feistel schemes with contracting functions.

2 Notation

Our notation is very similar to that used in [7] and [9]. We also follow the construction given in [9]. $[a, b]$ denotes the concatenation of strings a and b . An Unbalanced Feistel Scheme with Contracting Functions G_k^d is a Feistel scheme

with d rounds. At round j , we denote by f_j the round function from $(k - 1)n$ bits to n bits. On some input $[I^1, I^2, \dots, I^k]$, G_k^d produces an output denoted by $[S^1, S^2, \dots, S^k]$ by going through d rounds. At each round, the last $(k - 1)n$ bits of the round entry are used as an input to the round function f_j , which produces n bits. Those bits are xored to the first n bits of the round entry. Finally before going to round $j + 1$, the kn bit value is rotated by n bits.

We introduce the internal variable X^j : it is the only n -bit value which is modified at round j and which becomes the k coordinate of the internal state after j rounds. For example, we have:

$$\begin{aligned} X^1 &= I^1 \oplus f_1([I^2, \dots, I^k]), \\ X^2 &= I^2 \oplus f_2([I^3, \dots, I^k, X^1]), \\ X^3 &= I^3 \oplus f_3([I^4, \dots, I^k, X^1, X^2]), \\ &\dots \end{aligned}$$

The first round of G_k^d is represented in Fig. 1 below.

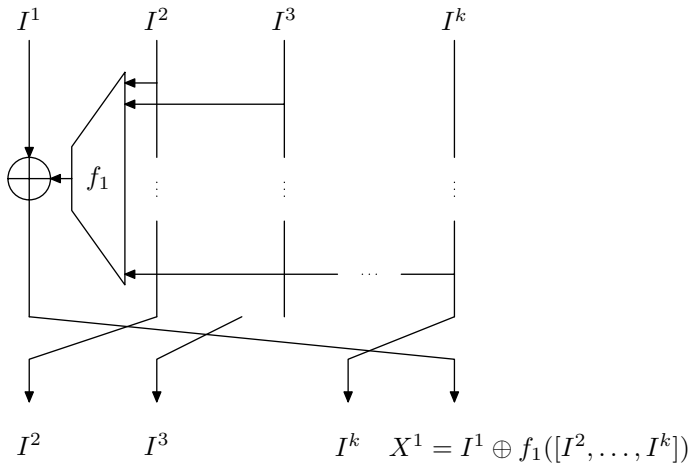


Fig. 1. First Round of G_k^d

3 Overview of the Attacks

We present several attacks that allow us to distinguish G_k^d from a random permutation. Depending on the number of rounds, it is possible to find some relations between the input variables and output variables. Those relations hold conditionally to equalities of some internal variables due to the structure of the Feistel scheme. Our attacks consist in using m plaintexts and ciphertexts tuples and in counting the number $\mathcal{N}_{G_k^d}$ of pairs of these tuples that satisfy the above relations. We then compare $\mathcal{N}_{G_k^d}$ with the equivalent number \mathcal{N}_{perm} if a random permutation is used instead of G_k^d . Our attack is successful, i.e. it is able to distinguish

G_k^d from a random permutation if the difference $|E(\mathcal{N}_{G_k^d}) - E(\mathcal{N}_{perm})|$ is much larger than the standard deviation σ_{perm} and than the standard deviation $\sigma_{G_k^d}$, where E denotes the expectancy function. More general cases of success are also given in the extended version of this paper [13].

In order to compute these values, we need to take into account the fact that the m^2 pairs obtained from the m plaintext/ciphertext tuples are not independent. However their mutual dependence is very small. To compute σ_{perm} and $\sigma_{G_k^d}$, we will use this well-known formula that we will call the ‘‘Covariance Formula’’:

$$V(\sum x_i) = \sum_i V(x_i) + \sum_{i < j} [E(x_i, x_j) - E(x_i)E(x_j)]$$

where the x_i are random variables.

We can note that for a small number of rounds $d < k$, a distinguishing attack is very easy to find. The output of G_k^d is $[S^1, S^2, \dots, S^k]$ which is equal to $[I^{d+1}, \dots, I^k, X^1, \dots, X^d]$. This shows that we can easily mount a KPA attack with one single message. We just have to test if the first coordinate of the output is equal to the coordinate of rank $d + 1$ of the input. This leads us to start investigating attacks for scheme with at least k rounds.

4 Generic Attacks When $k = 3$ and $3 \leq d \leq 6$

We first study schemes with $k = 3$ since this case is slightly different from the general case $k \geq 4$ and since it gives simple examples of what we will do. We have $[S_i^1, S_i^2, S_i^3] = G_3^d([I_i^1, I_i^2, I_i^3])$.

4.1 Attacks on 3 Rounds: G_3^3

G_3^3 : 3 rounds, CPA-1 with $m = 2$ messages. Let us choose $I_2^2 = I_1^2, I_2^3 = I_1^3$ and $I_2^1 \neq I_1^1$. Then the attack just tests if $S_1^1 \oplus S_2^1 = I_1^1 \oplus I_2^1$. This will occur with probability 1 if f is a G_3^3 , and with probability $\simeq \frac{1}{2^n}$ if f is a random permutation. So with three rounds there is a generic attack with two non-adaptive chosen queries and $O(1)$ computations.

G_3^3 : 3 rounds, KPA with $m \simeq 2^n$ messages. It is possible to transform this non-adaptive chosen plaintext attack into a known plaintext attack as follows. If we have $m \geq 2^n$ random inputs $[I_i^1, I_i^2, I_i^3]$, then (since $m^2 \geq 2^{2n}$) with a good probability we will have a collision $I_i^2 = I_j^2$ and $I_i^3 = I_j^3, i \neq j$. Then we test if $S_i^1 \oplus S_j^1 = I_i^1 \oplus I_j^1$. Now the attack requires $O(2^n)$ random queries and $O(2^n)$ computations.

4.2 Attacks on 4 Rounds: G_3^4

When the output $[I^1, I^2, I^3]$ is given, we have introduced the internal variable $X^1 = I^1 \oplus f_1([I^2, I^3])$ and the following conditions hold:

$$\begin{cases} I_i^2 = I_j^2 & \text{and } I_i^3 = I_j^3 & \Rightarrow X_i^1 \oplus X_j^1 = I_i^1 \oplus I_j^1 \\ I_i^3 = I_j^3 & \text{and } X_i^1 = X_j^1 & \Rightarrow S_i^1 \oplus S_j^1 = I_i^2 \oplus I_j^2 \\ X_i^1 = X_j^1 & \text{and } S_i^1 = S_j^1 & \Rightarrow S_i^2 \oplus S_j^2 = I_i^3 \oplus I_j^3 \\ S_i^1 = S_j^1 & \text{and } S_i^2 = S_j^2 & \Rightarrow S_i^3 \oplus S_j^3 = X_i^1 \oplus X_j^1 \end{cases}$$

The attack exploits the second condition. It proceeds as follows: we choose m messages such that $\forall i, I_i^3 = 0$ and $I_i^2 \neq I_j^2$ for all $i \neq j$. We then count $\mathcal{N}_{G_3^4}$ the number of pairs (i, j) with $i < j$ such that $I_i^2 \oplus I_j^2 = S_i^1 \oplus S_j^1$. For a random permutation, this condition appears only by chance. Thus we get:

$$\mathcal{N}_{perm} \simeq \frac{m^2}{2 \cdot 2^n} + O\left(\frac{m}{2^{\frac{n}{2}}}\right).$$

Here $O\left(\frac{m}{2^{\frac{n}{2}}}\right)$ denotes the standard deviation. This can be easily proved using the Covariance Formula, see Appendix A or full version of this article [13].

For G_3^4 , the equation $I_i^2 \oplus I_j^2 = S_i^1 \oplus S_j^1$ can occur at random with probability 2^{-n} or from the internal collision $X_i^1 = X_j^1$. Since I_i^3 is equal to zero for all i , we have $X_i^1 = I_i^1 \oplus f_1([I_i^2, 0])$. Since f_1 is a random function and the I^2 are pairwise distinct, the values $f_1([I_i^2, 0])$ and consequently the X_i^1 are uniformly distributed random variables. Consequently the internal collision $X_i^1 = X_j^1$ appears with probability 2^{-n} and we have:

$$\mathcal{N}_{G_3^4} \simeq \frac{m^2}{2^n} + O\left(\frac{m}{2^{\frac{n}{2}}}\right)$$

where $O\left(\frac{m}{2^{\frac{n}{2}}}\right)$ denotes the standard deviation (proof is given below). We can distinguish the two permutations when the difference between the mean values is larger than the standard deviation i.e. when $\frac{m^2}{2^n} \geq \frac{m}{2^{\frac{n}{2}}}$, i.e. for $m \geq 2^{\frac{n}{2}}$. This generic attack requires $O(2^{\frac{n}{2}})$ random queries and $O(2^{\frac{n}{2}})$ computations.

As explained previously, we can transform this attack in a known plaintext attack with $m \simeq 2^n$.

Proof of the Standard Deviation $\sigma_{G_3^4}$

We introduce the following random variables:

$$\begin{cases} \delta_{i,j} = 1 & \text{if } I_i^2 \oplus I_j^2 = S_i^1 \oplus S_j^1 \\ \delta_{i,j} = 0 & \text{otherwise.} \end{cases}$$

Since we have chosen all the I_i^3 equal to zero, we can say equivalently that $\delta_{i,j}$ is equal to one when $f_2([0, X_i^1]) = f_2([0, X_j^1])$. $\mathcal{N}_{G_3^4}$ is defined as $\sum_{i < j} \delta_{i,j}$ and it is easy to compute $E(\delta_{i,j}) = \frac{2}{2^n} - \frac{1}{2^{2n}}$. We now compute the variance $V(\delta_{i,j}) = E(\delta_{i,j}^2) - E(\delta_{i,j})^2 = E(\delta_{i,j}) - E(\delta_{i,j})^2 = \frac{2}{2^n} - \frac{5}{2^{2n}} + \frac{4}{2^{3n}} - \frac{1}{2^{4n}}$. We recall the Covariance Formula:

$$V\left(\sum_{i < j} \delta_{i,j}\right) = \sum_{i < j} V(\delta_{i,j}) + \sum_{i < j, k < l, (i,j) \neq (k,l)} [E(\delta_{i,j} \delta_{k,l}) - E(\delta_{i,j}) E(\delta_{k,l})].$$

We need to compute $Cov(i, j, k, l) = E(\delta_{i,j} \delta_{k,l}) - E(\delta_{i,j}) E(\delta_{k,l})$. Let us first consider the case, where i, j, k, l are pairwise distinct. We need to consider the influence of the equality $f_2([0, X_i^1]) = f_2([0, X_j^1])$ over the equality $f_2([0, X_k^1]) = f_2([0, X_l^1])$. It can only happen if $X_k^1 \neq X_l^1$ and if either $X_k^1 = X_i^1$ and $X_l^1 = X_j^1$ or $X_k^1 = X_j^1$ and $X_l^1 = X_i^1$. In that case we have also $X_i^1 \neq X_j^1$. This event happens with probability $(1 - \frac{1}{2^n}) \frac{2}{2^{2n}}$ and both equalities have a probability $\frac{1}{2^n}$ instead of $\frac{1}{2^{2n}}$. This gives a covariance equals to

$$\frac{2}{2^{3n}} - \frac{4}{2^{4n}} + \frac{2}{2^{5n}}.$$

The second case is if both equations are sharing an index, for example $i = k$. We need to consider the influence of the equality $f_2([0, X_i^1]) = f_2([0, X_j^1])$ over the equality $f_2([0, X_i^1]) = f_2([0, X_l^1])$. It can only happen if $X_i^1 \neq X_j^1$. This event happens with probability $(1 - \frac{1}{2^n}) \frac{1}{2^n}$ and both equalities have a probability $\frac{1}{2^n}$ instead of $\frac{1}{2^{2n}}$. This gives a covariance equals to

$$\frac{1}{2^{2n}} - \frac{2}{2^{3n}} + \frac{1}{2^{4n}}.$$

Consequently we have

$$V(\mathcal{N}_{G_3^4}) = \frac{m^2}{2^n} + O\left(\frac{m^3}{2^{2n}}\right) + O\left(\frac{m^4}{2^{3n}}\right)$$

Since m is smaller than 2^n , we get:

$$V(\mathcal{N}_{G_3^4}) \simeq \frac{m^2}{2^n} \text{ and } \sigma_{G_3^4} \simeq \frac{m}{2^{\frac{n}{2}}}.$$

4.3 Attacks on 5 Rounds: G_3^5

For 5 rounds, the internal variables are X^1 and $X^2 = I^2 \oplus f_2([I^3, X^1])$. We have the following conditions:

$$\begin{cases} I_i^2 = I_j^2 & \text{and } I_i^3 = I_j^3 & \Rightarrow X_i^1 \oplus X_j^1 = I_i^1 \oplus I_j^1 \\ I_i^3 = I_j^3 & \text{and } X_i^1 = X_j^1 & \Rightarrow X_i^2 \oplus X_j^2 = I_i^2 \oplus I_j^2 \\ X_i^1 = X_j^1 & \text{and } X_i^2 = X_j^2 & \Rightarrow S_i^1 \oplus S_j^1 = I_i^3 \oplus I_j^3 \\ X_i^2 = X_j^2 & \text{and } S_i^1 = S_j^1 & \Rightarrow S_i^2 \oplus S_j^2 = X_i^1 \oplus X_j^1 \\ S_i^1 = S_j^1 & \text{and } S_i^2 = S_j^2 & \Rightarrow S_i^3 \oplus S_j^3 = X_i^2 \oplus X_j^2 \end{cases}$$

The attack proceeds as follows: we choose m messages such that $\forall i, I_i^2 = 0, I_i^3 = 0$ and the I_i^1 values are pairwise distinct. Notice that this directly implies $X_i^1 \oplus X_j^1 = I_i^1 \oplus I_j^1$, so the X_i^1 values are pairwise distinct. Let \mathcal{N} be the number of pairs $(i, j), i < j$ such that $S_i^1 = S_j^1$ and $I_i^1 \oplus I_j^1 = S_i^2 \oplus S_j^2$. With a random permutation, these two conditions appear by chance and we have:

$$\mathcal{N}_{perm} \simeq \frac{m^2}{2 \cdot 2^{2n}} + O\left(\frac{m}{2^n}\right).$$

Here $O(\frac{m}{2^n})$ is the standard deviation. For a G_3^5 , $S_i^1 = S_j^1$ and $I_i^1 \oplus I_j^1 = S_i^2 \oplus S_j^2$ appear at random or as a consequence of $X_i^2 = X_j^2$ and $S_i^1 = S_j^1$. This gives:

$$\mathcal{N}_{G_3^5} \simeq \frac{m^2}{2^{2n}}.$$

We can distinguish the two permutations when the difference between the mean values is larger than the standard deviation i.e. when $\frac{m^2}{2^{2n}} \geq \frac{m}{2^n}$, or $m \geq 2^n$. Remark: here $m \leq 2^n$ since $I_i^2 = 0$ and $I_i^3 = 0$; so the attack will succeed when $m \simeq 2^n$.

As before this attack leads to a KPA attack with 2^{2n} messages. But there is a better attack as we can see now.

G_3^5 : 5 rounds, KPA with $m = 2^{\frac{3n}{2}}$ messages

For this attack, let \mathcal{N} be the number of pairs (i, j) , $i < j$, such that $I_i^3 \oplus I_j^3 = S_i^1 \oplus S_j^1$. For a random permutation, we have:

$$\mathcal{N}_{perm} \simeq \frac{m^2}{2 \cdot 2^n} + O\left(\frac{m}{\sqrt{2^n}}\right)$$

where $\frac{m}{\sqrt{2^n}}$ is the standard deviation, while for G_3^5 we obtain

$$\mathcal{N}_{G_3^5} \simeq \frac{m^2}{2 \cdot 2^n} + \frac{m^2}{2 \cdot 2^{2n}}.$$

We can distinguish the two permutations when the difference between the mean values is larger than the standard deviation i.e. when $\frac{m^2}{2^{2n}} \geq \frac{m}{\sqrt{2^n}}$, i.e. for $m \geq 2^{\frac{3}{2}n}$.

4.4 Attacks on 6 Rounds: G_3^6

For 6 rounds, the internal variables are X^1 , X^2 and $X^3 = I^3 \oplus f_3([X^1, X^2])$. We have the following conditions:

$$\left\{ \begin{array}{l} I_i^2 = I_j^2 \quad \text{and} \quad I_i^3 = I_j^3 \quad \Rightarrow \quad X_i^1 \oplus X_j^1 = I_i^1 \oplus I_j^1 \\ I_i^3 = I_j^3 \quad \text{and} \quad X_i^1 = X_j^1 \quad \Rightarrow \quad X_i^2 \oplus X_j^2 = I_i^2 \oplus I_j^2 \\ X_i^1 = X_j^1 \quad \text{and} \quad X_i^2 = X_j^2 \quad \Rightarrow \quad X_i^3 \oplus X_j^3 = I_i^3 \oplus I_j^3 \\ X_i^2 = X_j^2 \quad \text{and} \quad X_i^3 = X_j^3 \quad \Rightarrow \quad S_i^1 \oplus S_j^1 = X_i^1 \oplus X_j^1 \\ X_i^3 = X_j^3 \quad \text{and} \quad S_i^1 = S_j^1 \quad \Rightarrow \quad S_i^2 \oplus S_j^2 = X_i^2 \oplus X_j^2 \\ S_i^1 = S_j^1 \quad \text{and} \quad S_i^2 = S_j^2 \quad \Rightarrow \quad S_i^3 \oplus S_j^3 = X_i^3 \oplus X_j^3 \end{array} \right.$$

The attack proceeds as follows: we choose m messages such that $\forall i, I_i^3 = 0$. Let \mathcal{N} be the number of pairs (i, j) , $i < j$, such that $I_i^2 = I_j^2$ and $I_i^1 \oplus I_j^1 = S_i^1 \oplus S_j^1$. With a random permutation, we have:

$$\mathcal{N}_{perm} \simeq \frac{m^2}{2 \cdot 2^{2n}} + O\left(\frac{m}{2^n}\right)$$

where $O(\frac{m}{2^n})$ is the standard deviation. For a G_3^6 , since all the I_i^3 values are equal, $I_i^2 = I_j^2$ and $X_i^2 = X_j^2$ and $X_i^3 = X_j^3$ imply $I_i^1 \oplus I_j^1 = S_i^1 \oplus S_j^1$. We get

$$\mathcal{N}_{G_3^6} \simeq \frac{m^2}{2 \cdot 2^{2n}} + \frac{m^2}{2 \cdot 2^{3n}}.$$

We can distinguish the two permutations when the difference between the mean values is larger than the standard deviation i.e. when $\frac{m^2}{2^{3n}} \geq \frac{m}{2^n}$, i.e. for $m \geq 2^{2n}$.

We can obviously transform this CPA-1 attack into a KPA attack which will succeed as soon as we have $m \geq 2^{\frac{5n}{2}}$.

4.5 Experimental Results on G_3^6

We have implemented our CPA-1 and KPA attacks against G_3^6 for small values of n ($n = 6$ and $n = 8$). Our experimental values confirm the theoretical results. Our experiments were performed as follows:

- choose randomly an instance of G_3^6
- choose randomly a permutation: for this we use classical balanced Feistel scheme with a large number of rounds (more than 20)
- launch the attack in CPA-1 with $m = 2^{2n}$, in KPA with $m = 2^{3n}$ ($m = 2^{\frac{5n}{2}}$ also works).
- count the number of plaintext/ciphertext pairs satisfying the relations for the G_3^6 function and for the permutation
- iterate this procedure a large number of times (here 1000 times) to evaluate the mean values and the standard deviations
- compute the mean value and the standard deviation for both the G_3^6 function and the permutation

Table 2. Experimental results for KPA and CPA attacks on G_3^6

Attack	n	$\mathcal{N}_{G_3^6}$	\mathcal{N}_{perm}	$\mathcal{N}_{G_3^6} - \mathcal{N}_{perm}$	$\frac{m^2}{2 \cdot 2^{4n}}$	$\sigma_{G_3^6}$	σ_{perm}	$\frac{m}{\sqrt{2 \cdot 2^{\frac{3n}{2}}}}$
KPA	6	131006	129011	1995	2048	159	372	362.038
KPA	8	8388308	8355787	32521	32768	2862	2833	2896.309
CPA	6	2058	2009	49	32	45	44	45.254
CPA	8	32781	32601	180	128	178	185	182.019

Conclusion. Our experimental values for $\mathcal{N}_{G_3^6} - \mathcal{N}_{perm}$ are very close to the theoretical expected values ($\frac{m^2}{2 \cdot 2^{4n}}$ in KPA and $\frac{m^2}{2 \cdot 2^{3n}}$ in CPA-1). Similarly, our experimental values for ϵ_{perm} are very close to the theoretical expected values ($\frac{m}{\sqrt{2 \cdot 2^{\frac{3n}{2}}}}$ in KPA and $\frac{m}{\sqrt{2 \cdot 2^n}}$ in CPA-1). So these simulations confirm that we can distinguish G_3^6 from a random permutation with the complexity that we have given.

5 Generic Attacks When $k \geq 4$ and $k \leq d \leq 2k - 1$

5.1 Attacks for k Rounds

We first describe a CPA-1 attack with two messages. All the blocks of these two messages are equal to zero except the first one. We test if $I_1^1 \oplus I_2^1 = S_1^1 \oplus S_2^1$. Since $S^1 = X^1 = I^1 \oplus f_1([I^2, \dots, I^k])$, this will occur with probability 1 if f is a G_i^k , and with probability 2^{-n} if f is a random permutation. This gives the result.

As usual, we transform this attack into a KPA attack with $m = O(2^{\frac{n(k-1)}{2}})$. In that case with a high probability $I_i^2 = I_j^2, I_i^3 = I_j^3, \dots, I_i^k = I_j^k$. We test again if $S_i^1 \oplus S_j^1 = I_i^1 \oplus I_j^1$.

5.2 Attacks for $k + t$ Rounds, with $1 \leq t < k - 1$

In the CPA-1 attack, we choose $\forall i, I_i^{t+2} = \dots = I_i^k = 0$ and pairwise distinct $[I_i^1, \dots, I_i^t]$. This choice limits the maximal number of plaintext/ciphertext tuples to $m \leq 2^{(t+1)n}$. We then count the number \mathcal{N} of pairs $(i, j), i < j$, such that $I_i^{t+1} \oplus I_j^{t+1} = S_i^1 \oplus S_j^1$. For a random permutation, we have:

$$\mathcal{N}_{perm} \simeq \frac{m(m-1)}{2 \cdot 2^n} + O\left(\frac{m}{2^{\frac{n}{2}}}\right).$$

Here $O(\frac{m}{2^{\frac{n}{2}}})$ denotes the standard deviation. This can be easily proved using the Covariance Formula, see Appendix A or full version of this article [13].

For an unbalanced Feistel scheme, the preceding condition appears at random, but we also have the following property:

$$X_i^1 = X_j^1, \dots, X_i^t = X_j^t \Rightarrow S_i^1 \oplus S_j^1 = I_i^{t+1} \oplus I_j^{t+1}$$

since $S_i^1 = X^{t+1} = I^{t+1} \oplus f_{t+1}([I^{t+2}, \dots, I^k, X^1, \dots, X^t])$. This gives

$$\mathcal{N}_{G_k^{k+t}} \simeq \frac{m(m-1)}{2 \cdot 2^n} + \frac{m(m-1)}{2 \cdot 2^{tn}}, \text{ so } |E(\mathcal{N}_{G_k^{k+t}}) - E(\mathcal{N}_{perm})| \simeq \frac{m(m-1)}{2 \cdot 2^{tn}}.$$

Here again for $\mathcal{N}_{G_k^d}$, the standard deviation can be computed by using the Covariance Formula, as we have shown for G_3^4 (see full version of this article for the details [13]). Thus we distinguish when $\frac{m^2}{2^{tn}} \geq \frac{m}{2^{\frac{n}{2}}}$ i.e. when $m \geq 2^{(t-\frac{1}{2})n}$, which is compatible with the bound given above.

As usual, we are able transform this attack into a KPA attack which succeeds if $m \geq 2^{(\frac{k+t-2}{2})n}$.

5.3 Attacks for $2k - 1$ Rounds

In that case we can only mount a KPA attack. We consider the following KPA attack: let \mathcal{N} be the number of pairs $(i, j), i < j$, such that $I_i^k \oplus I_j^k = S_i^1 \oplus S_j^1$. For a random permutation, we have $\mathcal{N}_{perm} \simeq \frac{m(m-1)}{2 \cdot 2^n} + O(\frac{m}{\sqrt{2^n}})$ and for an

unbalanced Feistel scheme, $\mathcal{N}_{G_k^{2k-1}} \simeq \frac{m(m-1)}{2 \cdot 2^n} + \frac{m(m-1)}{2 \cdot 2^{(k-1)n}}$, since $I_i^k \oplus I_j^k = S_i^1 \oplus S_j^1$ is also implied by the following equations: $X_i^1 = X_j^1, X_i^2 = X_j^2, \dots, X_i^{k-1} = X_j^{k-1}$. This is because $S^1 = X^k = I^k \oplus f_{2k-1}([X^1, \dots, X^{k-1}])$. Thus we can distinguish when $\frac{m^2}{2 \cdot 2^{(k-1)n}} \geq \frac{m}{\sqrt{2^n}}$. This gives $m \geq 2^{(k-\frac{3}{2})n}$.

We can remark that for more than $2k$ rounds we will have to proceed with different attacks, since $X_i^1 = X_j^1, \dots, X_i^k = X_j^k$ implies $i = j$ because we have a permutation.

6 Attacks with More Than 2^{kn} Computations

Until now we have studied Unbalanced Feistel schemes with random functions. In practice, for example in designing block ciphers we need to consider generators of pseudo-random permutations. In this section, we will describe attacks against a generator of permutations (and not only against a single permutation randomly generated by a generator of permutations), i.e. we will be able to study several permutations generated by the generator. This allows more than 2^{kn} computations.

Let G be a “ G_k^d generator”, i.e. from a binary string K , G generates a d round unbalanced Feistel permutation G_k^d . Let G' be a truly random permutation generator, i.e. from a string K , G' generates a truly random permutation G'_K of B_{kn} . Let G'' be a truly random even permutation generator, i.e. from a string K , G'' generates a truly random permutation G''_K of A_{kn} , with A_{kn} being the group of all the permutations of $\{0, 1\}^{kn} \rightarrow \{0, 1\}^{kn}$ with even signature. We are looking for attacks that distinguish G from G' , and also for attacks that will distinguish G from G'' .

Adversarial model: An attacker can choose some strings K_1, \dots, K_f , can ask for some inputs $[I^1, \dots, I^k]$, and can ask for some $G_{K_\alpha}[I^1, \dots, I^k]$ (with K_α being one of the K_i). Here the attack is more general than in the previous sections, since the attacker can have access to many different permutations generated by the same generator.

Adversarial goal: The aim of the attacker is to distinguish G from G' (or from G'') with a high probability and with a complexity as small as possible.

6.1 Brute Force Attacks

A possible attack is an exhaustive search for the d round functions f_1, \dots, f_d from $\{0, 1\}^{(k-1)n}$ to $\{0, 1\}^n$ that have been used in the unbalanced Feistel construction. This attack always exists, but since we have $2^{d \cdot n \cdot 2^{(k-1)n}}$ possibilities for f_1, \dots, f_d , this attack requires about $2^{d \cdot n \cdot 2^{(k-1)n}}$ computations and about $\frac{d}{k} \cdot 2^{(k-1)n}$ random queries but only for one permutation of the generator. This attacks means that an adversary with infinite computing power will be able to distinguish G_k^d from a random permutation (or from a truly random permutation with even signature) when $m \geq \frac{d}{k} \cdot 2^{(k-1)n}$.

6.2 Attack by the Signature

Theorem 1. *Let Ψ be an unbalanced Feistel permutation on $\{0, 1\}^{\alpha+\beta} \rightarrow \{0, 1\}^{\alpha+\beta}$ with round functions of $\{0, 1\}^\beta \rightarrow \{0, 1\}^\alpha$. Then if $\alpha \geq 2$ and $\beta \geq 1$, Ψ has an even signature.*

The proof of this theorem is quite similar to the proof in the case of a symmetric Feistel scheme [11,3]. However the fact that $\alpha \geq 2$ changes a few things. Consequently a complete proof is included in the full version [13], available from the authors.

Let f be a permutation from kn bits to kn bits. Then using $O(2^{kn})$ computations on the 2^{kn} input/output values of f , we can compute the signature of f . To achieve this we just compute all the cycles c_i of f , $f = \prod_{i=1}^{\alpha} c_i$ and use the formula:

$$\text{signature}(f) = \prod_{i=1}^{\alpha} (-1)^{\text{length}(c_i)+1}.$$

The consequence is that it is possible to distinguish G a generator of G_k^d from a generator of truly random permutations from kn bits to kn bits after $O(2^{kn})$ computations on $O(2^{kn})$ input/output values.

Remark: To compute the signature of a permutation g we need however to know all the input/outputs of g (or all of them minus one, since the last one can be found from the others if g is a permutation).

6.3 Attacks of G_k^d Generators When $d = 2k$

Let μ be the number of permutations that we will use. After $2k$ rounds, the output is given by $[S^1, S^2, \dots, S^k] = [X^{k+1}, X^{k+2}, \dots, X^{2k}]$ where we have $X^{k+1} = X^1 \oplus f_{k+1}([X^2, \dots, X^k])$. Remember that $X^1 = I^1 \oplus f_1([I^2, \dots, I^k])$. Let us describe the KPA attack which concentrates on $S^1 = X^{k+1}$. Let \mathcal{N} be the number of pairs (i, j) , $i < j$, such that

$$I_i^2 = I_j^2, \dots, I_i^k = I_j^k, \quad X_i^{k+1} \oplus X_j^{k+1} = I_i^1 \oplus I_j^1. \tag{1}$$

There we have necessary $I_i^1 \neq I_j^1$ and $X_i^1 \neq X_j^1$. When we are testing random permutations, $\mathcal{N}_{perm} \simeq \mu \cdot \frac{m^2}{2 \cdot 2^{kn}} + O(\sqrt{\mu} \cdot \frac{m}{2^{\frac{kn}{2}}})$. For G_k^k , since $I_i^2 = I_j^2, \dots, I_i^k = I_j^k, X_i^2 = X_j^2, \dots, X_i^k = X_j^k$ imply (1) we have:

$$\mathcal{N}_{G_k^d} = \mu \cdot \frac{m^2}{2 \cdot 2^{kn}} + \mu \cdot \frac{m^2}{2 \cdot 2^{(2k-2)n}}.$$

Thus we can distinguish the two generators when: $\mu \cdot \frac{m^2}{2^{(2k-2)n}} \geq \sqrt{\mu} \cdot \frac{m}{2^{\frac{kn}{2}}}$, or when $\mu \cdot m \geq 2^{(3k-4)n}$. When $m = 2^{kn}$, we find $\mu = 2^{(k-4)n}$ and $\mu \cdot m = 2^{(2k-4)n}$.

6.4 Attacks G_k^d Generators for d Rounds with $d \geq 2k$

It is possible to generalize the attack given above for any $d \geq 2k$. We give here only the main ideas. We concentrate the attack on X^{d-k+1} . In the constraints, there are d conditions and $d - k$ internal variables X^i . We choose conditions number $k, 2k, \dots$, until we get $\xi = \lfloor \frac{d}{k} \rfloor$ conditions. This gives ξ (internal or external) $\cdot (k - 1)$ -multiple equations. When they are satisfied, we have:

1. One equation between the input and output variables.
2. φ equations between the output variables where

$$\varphi = (k - 1) - \left(d - \left\lfloor \frac{d}{k} \right\rfloor k \right) = (k - 1) - (d \bmod k)$$

We have μ permutations and the attack proceeds as follows: let \mathcal{N} be the number of pairs (i, j) , $i < j$, such that these $\varphi + 1$ equations are satisfied. When we are testing a permutation generator, we have

$$\mathcal{N}_{perm} = \mu \cdot \frac{m(m - 1)}{2 \cdot 2^{(\varphi+1)n}} + O(\sqrt{\mu} \cdot \frac{m}{2^{(\frac{\varphi+1}{2})n}}).$$

With a G_k^d , the $\xi(k - 1)$ -multiples equations imply the $\varphi + 1$ equations described above. This shows that

$$\mathcal{N}_{G_k^d} = \mu \cdot \frac{m(m - 1)}{2 \cdot 2^{(\varphi+1)n}} + \mu \cdot \frac{m(m - 1)}{2 \cdot 2^{(k-1)n}}.$$

We get the condition:

$$\begin{aligned} \mu \cdot \frac{m^2}{2^{(k-1)n}} &\geq \sqrt{\mu} \cdot \frac{m}{2^{(\frac{\varphi+1}{2})n}}, \\ \mu \cdot m^2 &\geq 2^{(2(k-1)\xi - \varphi - 1)n}. \end{aligned}$$

For the maximal value $m = 2^{kn}$, we find $\mu = 2^{(2(k-1)\xi - \varphi - 2k - 1)n}$ and the complexity is $\lambda = \mu \cdot m = 2^{(2(k-1)\xi - \varphi k - 1)n}$. Thus we can write

$$\lambda = 2^{(2(k-1)\lfloor \frac{d}{k} \rfloor + (d \bmod k) - 2k)n} = 2^{(d + (k-2)\lfloor \frac{d}{k} \rfloor - 2k)n}.$$

7 Conclusion

Until now, attacks and proofs of security on contracting unbalanced Feistel Schemes have not received much attention. There are much more papers on classical Feistel schemes and even attacks on expanding unbalanced Feistel schemes have been more studied than attacks on contracting unbalanced Feistel schemes. This may be not justified since contracting Feistel schemes seem to have very good security properties. For example, to avoid all known generic attacks with the number of messages less than 2^{kn} (where kn is the number of bits of the input and the output) with these schemes, we need only $2k$ rounds (if $k \geq 4$)

Table 3. Results on G_3^d . For more than 7 rounds more that one permutation is needed or more than 2^{3n} computations are needed in the best known attacks to distinguish from a random permutation with an even signature.

	KPA	CPA-1 ^a
G_3^1	1	1
G_3^2	1	1
G_3^3	2^n	2
G_3^4	2^n	$2^{n/2}$
G_3^5	$2^{3n/2}$	2^n
G_3^6	$2^{5n/2}$	2^{2n}
G_3^7	2^{3n}	2^{3n}
G_3^8	2^{4n}	2^{4n}
G_3^9	2^{6n}	2^{6n}
G_3^{10}	2^{7n}	2^{7n}
G_3^{11}	2^{8n}	2^{8n}
G_3^{12}	2^{10n}	2^{10n}
$G_3^d, d \geq 12$	$2^{(d+\lfloor \frac{d}{3} \rfloor - 6)n}$	$2^{(d+\lfloor \frac{d}{3} \rfloor - 6)n}$

^a Here we do not show CPA-2, CPCA-1 and CPCA-2 since for G_3^d , no better attacks are found compared with CPA-1.

Table 4. Results on G_k^d for any $k \geq 4$. For more than $2k$ rounds more that one permutation is needed or more than $2^{(2k-4)n}$ computations are needed in the best known attacks to distinguish from a random permutation with an even signature.

	KPA	CPA-1 ^a
$G_k^d, 1 \leq d \leq k - 1$	1	1
G_k^k	$2^{\frac{n(k-1)}{2}}$	2
G_k^{k+1}	$2^{\frac{n(k-1)}{2}}$	$2^{\frac{n}{2}}$
G_k^{k+2}	$2^{\frac{k}{2}n}$	$2^{\frac{3}{2}n}$
G_k^{k+3}	$2^{(\frac{k+1}{2})n}$	$2^{\frac{5}{2}n}$
$G_k^{k+i}, 1 \leq i < k$	$2^{(\frac{k+i-2}{2})n}$	$2^{(\frac{2i-1}{2})n}$
G_k^{2k}	$2^{(2k-4)n}$	$2^{(2k-4)n}$
$G_k^d, d \geq 2k$	$2^{(d+(k-2)\lfloor \frac{d}{k} \rfloor - 2k)n}$	$2^{(d+(k-2)\lfloor \frac{d}{k} \rfloor - 2k)n}$

^a Here we do not show CPA-2, CPCA-1 and CPCA-2 since for G_k^d , no better attacks are found compared with CPA-1.

or 7 rounds (if $k = 3$). So each bit will be changed only 2 times (if $k \geq 4$) unlike with balanced Feistel schemes where 3 changes (i.e. 6 rounds) are necessary and unlike expanding unbalanced Feistel schemes where much more changes are needed [4,11,14].

Storing a random function of $(k - 1)n$ bits to n bits requires a large memory and this may be a practical disadvantage of G_k^d compared with balanced

Feistel schemes or Feistel schemes with expanding functions. However if a function generator is used to generate pseudo-random functions, this may not be a problem.

There are still many open problems on contracting unbalanced Feistel schemes. Naor and Reingold have shown a very nice security result [9]: we have security until the birthday bound when we use pairwise independent functions for the first and the last rounds. However, if we do not use such first and last rounds, the exact security is still an open problem and even the birthday security bound is not proved yet.

In conclusion, contracting unbalanced Feistel schemes seem to be one of the best designs for permutation generators. In this paper, we have presented attacks on these schemes with fewer than $2k$ rounds.

References

1. William Aiello and Ramarathnam Venkatesan. Foiling Birthday Attacks in Length-Doubling Transformations - Benes: A Non-Reversible Alternative to Feistel. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 307–320. Springer-Verlag, 1996.
2. Don Coppersmith. Luby-Rackoff: Four rounds is not enough. Technical Report RC20674, IBM Research Report, december 1996.
3. Shimon Even and Oded Goldreich. Des-like functions can generate the alternating group. *IEEE Transactions on Information Theory*, 29(6):863–865, 1983.
4. Charanjit S. Jutla. Generalized Birthday Attacks on Unbalanced Feistel Networks. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 186–199. Springer-Verlag, 1998.
5. Lars R. Knudsen. DEAL - A 128-bit Block Cipher. Technical Report 151, University of Bergen, Department of Informatics, Norway, february 1998.
6. Lars R. Knudsen and Vincent Rijmen. On the Decorrelated Fast Cipher (DFC) and Its Theory. In Lars R. Knudsen, editor, *Fast Software Encryption - FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 81–94. Springer-Verlag, 1999.
7. Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
8. Stefan Lucks. Faster Luby-Rackoff Ciphers. In Dieter Gollman, editor, *Fast Software Encryption - FSE '96*, volume 1039 of *Lecture Notes in Computer Science*, pages 189–203. Springer-Verlag, 1996.
9. Moni Naor and Omer Reingold. On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. *J. Cryptology*, 12(1):29–66, 1999.
10. Jacques Patarin. New Results on Pseudorandom Permutation Generators Based on the DES Scheme. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 301–312. Springer-Verlag, 1991.
11. Jacques Patarin. Generic Attacks on Feistel Schemes. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 222–238. Springer-Verlag, 2001.
12. Jacques Patarin. Security of Random Feistel Schemes with 5 or More Rounds. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122. Springer-Verlag, 2004.

13. Jacques Patarin, Valérie Nachev, and Côme Berbain. Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions, Extended Version. *available from the authors*, 2006.
14. Jacques Patarin, Valérie Nachev, and Côme Berbain. Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions. *available from the authors*, 2006.
15. Bruce Schneier and John Kelsey. Unbalanced Feistel Networks and Block Cipher Design. In Dieter Gollmann, editor, *Fast Software Encryption – FSE '96*, volume 1039 of *Lecture Notes in Computer Science*, pages 121–144. Springer-Verlag, 1996.

A Computation of the Variance for Random Permutations

In this section, we compute the value of the variance when we are testing a random permutation and we want to distinguish it from a G_k^{k+t} , $1 \leq t \leq k-1$. The input is $[I^1, \dots, I^k]$ and the output is $[S^1, \dots, S^k]$. We want to compute \mathcal{N}_{perm} which is the number of (i, j) , $i < j$ satisfying the relation $I_i^{t+1} \oplus S_i^1 = I_j^{t+1} \oplus S_j^1$. We have the condition $\forall i, I_i^{t+2} = I_i^{t+3} = \dots = I_i^k = 0$. This implies that $m \leq 2^{(t+1)n}$. We introduce the following random variables:

$$\begin{cases} \delta_{i,j} = 1 & \text{if } I_i^{t+1} \oplus S_i^1 = I_j^{t+1} \oplus S_j^1 \\ \delta_{i,j} = 0 & \text{otherwise} \end{cases}$$

Then $\mathcal{N}_{perm} = \sum_{i < j} \delta_{i,j}$ and $E(\delta_{i,j}) = Pr_{f \in_R B_{kn}} [I_i^{t+1} \oplus S_i^1 = I_j^{t+1} \oplus S_j^1]$.

Notice that if $m \ll 2^n$, we may assume that the I^{t+1} values are pairwise distinct (or are all equal) and if $m \geq 2^n$, we may assume that each element of $\{0, 1\}^n$ is reached by about $\frac{m}{2^n}$ values of I_i^{t+1} (in CPA-1, we can choose m to be a multiple of 2^n and each element of $\{0, 1\}^n$ is reached by exactly $\frac{m}{2^n}$ values of I_i^{t+1} . It is also possible to choose that I_i^{t+1} are random values). If $I_i^{t+1} = I_j^{t+1}$, $E(\delta_{i,j}) = Pr_{f \in_R B_{kn}} [S_i^1 = S_j^1] = \frac{2^{(k-1)n} - 1}{2^{kn} - 1} \simeq \frac{1}{2^n}$. and if $I_i^{t+1} \neq I_j^{t+1}$, $E(\delta_{i,j}) = \frac{2^{(k-1)n}}{2^{kn} - 1} \simeq \frac{1}{2^n}$. This gives us the average value:

$$E(\mathcal{N}_{perm}) \simeq \frac{m(m-1)}{2 \cdot 2^n} + o\left(\frac{m}{2^{(k+\frac{1}{2})n}}\right).$$

We now compute the variance $V(\delta_{i,j}) = E(\delta_{i,j}^2) - E(\delta_{i,j})^2 = E(\delta_{i,j}) - E(\delta_{i,j})^2$. If $I_i^{t+1} = I_j^{t+1}$, $V(\delta_{i,j}) = \frac{1}{2^n} \cdot \frac{1}{1 - \frac{1}{2^{kn}}} - \frac{1}{2^{kn} - 1} - \left(\frac{1}{2^n} \cdot \frac{1}{1 - \frac{1}{2^{kn}}} - \frac{1}{2^{kn} - 1}\right)^2$. And if $I_i^{t+1} \neq I_j^{t+1}$, $V(\delta_{i,j}) = \frac{1}{2^n} \cdot \frac{1}{1 - \frac{1}{2^{kn}}} - \left(\frac{1}{2^n} \cdot \frac{1}{1 - \frac{1}{2^{kn}}}\right)^2$. Finally $V(\delta_{i,j}) \simeq \frac{1}{2^n} \left(1 - \frac{1}{2^n}\right)$ and

$$\sum_{i < j} V(\delta_{i,j}) \simeq \frac{m(m-1)}{2} \cdot \frac{1}{2^n} \left(1 - \frac{1}{2^n}\right).$$

We recall the formula:

$$V(\mathcal{N}_{perm}) = V\left(\sum_{i < j} \delta_{i,j}\right) = \sum_{i < j} V(\delta_{i,j}) + \sum_{i < j, p < l, (i,j) \neq (p,l)} [E(\delta_{i,j} \delta_{p,l}) - E(\delta_{i,j}) E(\delta_{p,l})]$$

The second term is the covariance term. We will see that

$$V(\mathcal{N}_{perm}) = \frac{m(m-1)}{2 \cdot 2^n} + O\left(\frac{m^2}{2^{2n}}\right) + O\left(\frac{m^4}{2^{2n} \cdot 2^{(2k-1)n}}\right) + O\left(\frac{m^3}{2^{2n} \cdot 2^{(k-1)n}}\right)$$

where the two first terms correspond to the sum of the variance of $\delta_{i,j}$, the third term corresponds to the covariance of four distinct indexes (i, j, k, l) , and the last term corresponds to the covariance of 4-tuples of indexes with one in common, like for example (i, j, i, l) . Therefore, for m larger than 2^n but smaller than 2^{kn} , we have as claimed

$$V(\mathcal{N}_{perm}) = \frac{m(m-1)}{2 \cdot 2^n} + o\left(\frac{m^2}{2^n}\right) \simeq \frac{m^2}{2 \cdot 2^n}.$$

In order to exactly compute the covariance term, we can separate the computation into several cases. Here we only study the main case, i.e. we suppose that i, j, p, l are pairwise distinct and that $I_i^{t+1} \neq I_j^{t+1}$, $I_p^{t+1} \neq I_l^{t+1}$ and $I_i^{t+1} \oplus I_j^{t+1} \oplus I_p^{t+1} \oplus I_l^{t+1} \neq 0$. For all other cases, computation is similar and is included in the full version of this paper [13].

To compute this probability we need to count the total number A of possibilities for the outputs $[S_i^1, \dots, S_i^k]$, $[S_j^1, \dots, S_j^k]$, $[S_p^1, \dots, S_p^k]$ and $[S_l^1, \dots, S_l^k]$. Since we are using a permutation, we have $A = 2^{kn} \cdot (2^{kn} - 1) \cdot (2^{kn} - 2) \cdot (2^{kn} - 3)$.

We also have to compute B the number of outputs $[S_i^1, \dots, S_i^k]$, $[S_j^1, \dots, S_j^k]$, $[S_p^1, \dots, S_p^k]$ and $[S_l^1, \dots, S_l^k]$ satisfying the above relations in the case we consider. For $[S_i^1, \dots, S_i^k]$, there are 2^{kn} possibilities. When this output is fixed, $S_j^1 = S_i^1 \oplus I_i^{t+1} \oplus I_j^{t+1}$. Thus there are $2^{(k-1)n}$ possibilities for $[S_j^1, \dots, S_j^k]$. Now we have to fix $[S_i^1, \dots, S_i^k]$ and $[S_j^1, \dots, S_j^k]$. There are 5 cases that we are going to study now. If $S_p^1 = S_i^1 \oplus I_p^{t+1} \oplus I_l^{t+1}$, then $S_p^1 \neq S_i^1$, $S_p^1 \neq S_l^1$ and $S_l^1 = S_i^1$. Thus we have $2^{(k-1)n} \cdot (2^{(k-1)n} - 1)$ possibilities for $[S_p^1, \dots, S_p^k]$ and $[S_l^1, \dots, S_l^k]$. Then we consider the case where $S_p^1 = S_j^1 \oplus I_p^{t+1} \oplus I_l^{t+1}$. This case is different from the previous one since $S_i^1 \neq S_j^1$. We get again $2^{(k-1)n} \cdot (2^{(k-1)n} - 1)$ possibilities for $[S_p^1, \dots, S_p^k]$ and $[S_l^1, \dots, S_l^k]$. If $S_p^1 = S_i^1$ or if $S_p^1 = S_j^1$, there are $(2^{(k-1)n} - 1) \cdot 2^{(k-1)n}$ possibilities for $[S_p^1, \dots, S_p^k]$ and $[S_l^1, \dots, S_l^k]$. The last case is when we have eliminated the previous cases. This gives $(2^n - 4) \cdot 2^{(k-1)n} \cdot 2^{(k-1)n}$ possibilities for $[S_p^1, \dots, S_p^k]$ and $[S_l^1, \dots, S_l^k]$. Finally $B = 2^{(4k-2)n} \cdot (1 - \frac{4}{2^{kn}})$. Consequently, since $E(\delta_{i,j} \delta_{p,l}) = \frac{B}{A}$ we get:

$$E(\delta_{i,j} \delta_{p,l}) - E(\delta_{i,j})E(\delta_{p,l}) = \frac{1}{2^{2n}} \left(-\frac{2}{2^{2kn}} + O\left(\frac{1}{2^{3kn}}\right) \right).$$

Finally these terms of covariance are equal to $\frac{-2m^4}{4 \cdot 2^{2n} \cdot 2^{2kn}} \leq O\left(\frac{m^4}{2^{2n} \cdot 2^{(2k-1)n}}\right)$ as claimed.

New Cryptanalytic Results on IDEA

Eli Biham^{1,*}, Orr Dunkelman^{1,*}, and Nathan Keller^{2,**}

¹Computer Science Department, Technion,
Haifa 32000, Israel

{biham, orrd}@cs.technion.ac.il

²Einstein Institute of Mathematics, Hebrew University,
Jerusalem 91904, Israel
nkeller@math.huji.ac.il

Abstract. IDEA is a 64-bit block cipher with 128-bit keys introduced by Lai and Massey in 1991. IDEA is one of the most widely used block ciphers, due to its inclusion in several cryptographic packages, such as PGP and SSH. The cryptographic strength of IDEA relies on a combination of three incompatible group operations – XOR, addition and modular multiplication. Since its introduction in 1991, IDEA has withstood extensive cryptanalytic effort, but no attack was found on the full variant of the cipher.

In this paper we present the first known non-trivial relation that involves all the three operations of IDEA. Using this relation and other techniques, we devise a linear attack on 5-round IDEA that uses 2^{19} known plaintexts and has a time complexity of 2^{103} encryptions. By transforming the relation into a related-key one, a similar attack on 7.5-round IDEA can be applied with data complexity of $2^{43.5}$ known plaintexts and a time complexity equivalent to $2^{115.1}$ encryptions. Both of the attacks are by far the best known attacks on IDEA

1 Introduction

The International Data Encryption Algorithm (IDEA) is a 64-bit, 8.5-round block cipher with 128-bit keys proposed by Lai and Massey in 1991 [20]. Due to its inclusion in several cryptographic packages, such as PGP and SSH, IDEA is one of the most widely used block ciphers. Since its introduction, IDEA resisted intensive cryptanalytic efforts [1, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 21, 22, 24]. The best published chosen-plaintext attack on IDEA is an attack on 5-round IDEA that requires 2^{24} chosen plaintexts, and has time complexity of 2^{126} encryptions [12]. The best published related-key attack is an attack on 6.5-round IDEA that requires $2^{57.8}$ chosen plaintexts encrypted under four related keys and has time complexity of $2^{88.1}$ encryptions [5]. Along with the attacks on reduced-round variants, several weak-key classes for the entire IDEA were found. The largest weak key class (identified by a boomerang technique) contains 2^{64} keys,

* This work was supported in part by the Israel MOD Research and Technology Unit.

** The research presented in this paper was supported by the Adams fellowship.

and the membership test requires 2^{16} adaptive chosen plaintexts and ciphertexts and has a time complexity of 2^{16} encryptions [6].

The cryptographic strength of IDEA relies on the combination of three incompatible group operations: bitwise XOR, modular addition in $Z_{2^{16}}$, and modular multiplication in $GF(2^{16} + 1)$ where 0 is replaced by 2^{16} . All the three operations are essential for the security of the cipher. Indeed, if the multiplication is removed, then the cipher can be broken easily by examining the least significant bits of the words during the encryption. If the XOR is removed, then the cipher is affine over addition in $Z_{2^{16}}$, and hence, is easily breakable using only few known plaintexts. In [7, 26] it is shown that if the addition is removed then the cipher can be easily broken using multiplicative differentials.

In this paper we present the first known non-trivial relation that involves all the three different operations of IDEA. More precisely, we show that for the MA transformation of IDEA, that is composed of additions and multiplications, there exists an XOR differential with a non-trivial probability.

We use our new relation to devise several new attacks on IDEA based on various attack techniques: First, we devise linear-type attacks on reduced-round variants of IDEA that are similar to the attacks presented in [12, 16, 24]. The attacks are based on constructing linear approximations with bias $1/2$ that relates the least significant bits of some words during the encryption process. We use our relation, along with differential techniques and partial key guessing, to improve the basic technique presented in [16, 24] and to establish the best known attack on 5-round IDEA. Our attack requires only 2^{19} known plaintexts and the time complexity is equivalent to 2^{103} encryptions. Both the data and the time complexities are smaller than the respective complexities of all the previously known attacks on 4.5 or 5 rounds of IDEA. Our attack also has a relatively small memory complexity, unlike the 5-round attack in [12]. We also devise realistic attacks on variants of IDEA with a small number of rounds: A distinguishing attack on 2.5-round IDEA requiring 2^{18} chosen plaintexts and time complexity of 2^{18} encryptions, and an attack on 3-round IDEA with data complexity of 2^{19} chosen plaintexts and time complexity of about $2^{48.5}$ encryptions. Both of the attacks are better in some of the parameters than all the known attacks on the respective variants of IDEA.

We also show how to use the same relation in the related-key model. Using two related keys, we are able to extend the linear property by 2.5 rounds. This gives rise to a 7.5-round attack on IDEA requiring $2^{43.5}$ known plaintexts and a time complexity of $2^{115.1}$ encryptions. It is also possible to use our new relation to improve the previously best known related-key attack on IDEA, using the related-key rectangle technique. These improvements can be used to construct a 7-round related-key rectangle attack on IDEA with data complexity of 2^{65} related-key chosen plaintexts and time complexity of $2^{104.2}$ 7-round IDEA encryptions. The complexities of the new attacks, along with selected previously known attacks, are summarized in Table 1.

Table 1. Selected Known Attacks on IDEA and Our New Results

Rounds	Attack Type	Complexity		# of Affected Keys	Source
		Data	Time		
2	Differential	2^{10} CP	2^{42}	all	[21]
2.5	Differential	2^{10} CP	2^{106}	all	[21]
3	Differential-Linear	2^{29} CP	2^{44}	all	[8]
3.5	Linear	103 KP/CP	2^{97}	all	[16]
3.5	Square	2^{22} CP	2^{66}	all	[16]
4	Impossible Differential	2^{37} CP	2^{70}	all	[1]
4	Linear	114 KP	2^{114}	all	[24]
4	Square	2^{23} CP	2^{98}	all	[16]
4.5	Impossible Differential	2^{64} CP	2^{112}	all	[1]
5	Meet-in-the-Middle Attack	2^{24} CP	2^{126}	all	[12]
6.5	Related-Key Rectangle	$2^{59.8}$ RK-CP	$2^{88.1}$	all	[5]
2.5 [†]	Linear	2^{18} CP	2^{18}	all	Section 4.1
3	Linear	2^{19} CP	$2^{48.5}$	all	Section 4.2
4.5	Linear	16 CP	2^{103}	all	Section 4.3
5	Linear	2^{19} KP	2^{103}	all	Section 4.3
7.5	Related-Key Linear	$2^{43.5}$ RK-KP	$2^{115.1}$	all	Section 5
7	Related-Key Rectangle	2^{65} RK-CP	$2^{104.2}$	all	Appendix A

KP – Known plaintext, CP – Chosen plaintext, RK – Related key,
Time complexity is measured in encryption units.

[†] – Distinguishing attack.

We expect that the new relation can also be used to improve other attacks on IDEA, as well as attacks on other block ciphers that use the same operations, e.g., the MESH family of block ciphers [23].

The paper is organized as follows: In Section 2, we briefly describe the structure of IDEA. In Section 3 we present the new relation between the operations of IDEA. In Section 4 we present the new attack on 5-round IDEA. In Section 5 we transform this attack into a 7.5-round related-key attack on IDEA. Appendix A suggests a related-key rectangle attack on 7-round IDEA. Finally, Section 6 summarizes the paper.

2 Description of IDEA and the Notations Used in the Paper

IDEA [20] is a 64-bit, 8.5-round block cipher with 128-bit keys. It uses a composition of XOR operations, additions modulo 2^{16} , and multiplications over $GF(2^{16} + 1)$.

Every round of IDEA is composed of two layers. The round input of round i is composed of four 16-bit words denoted by $(X_1^i, X_2^i, X_3^i, X_4^i)$. In the first layer, denoted by KA , the first and the fourth words are multiplied by subkey words (mod $2^{16} + 1$) where 0 is replaced by 2^{16} , and the second and the third words are added to subkey words in (mod 2^{16}). The intermediate values after this

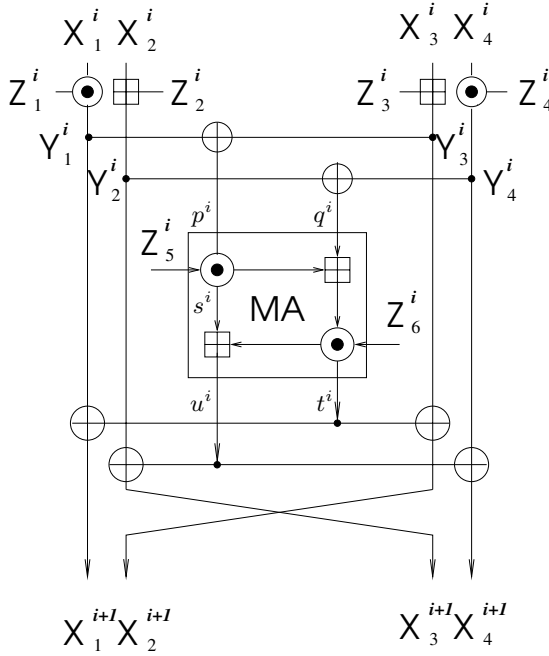


Fig. 1. One Round of IDEA

half-round are denoted by $(Y_1^i, Y_2^i, Y_3^i, Y_4^i)$. Formally, let Z_1^i, Z_2^i, Z_3^i , and Z_4^i be the four subkey words, then

$$Y_1^i = Z_1^i \odot X_1^i; \quad Y_2^i = Z_2^i \boxplus X_2^i; \quad Y_3^i = Z_3^i \boxplus X_3^i; \quad Y_4^i = Z_4^i \odot X_4^i$$

Then, $(p^i, q^i) = (Y_1^i \oplus Y_3^i, Y_2^i \oplus Y_4^i)$ enters the second layer, a structure composed of multiplications and additions denoted by MA . We denote the two output words of the MA transformation by (u^i, t^i) . Denoting the subkey words that enter the MA function by Z_5^i and Z_6^i ,

$$u^i = (p^i \odot Z_5^i) \boxplus t^i; \quad t^i = (q^i \boxplus (p^i \odot Z_5^i)) \odot Z_6^i$$

Another notation we use in the attack refers to an intermediate value in the MA layer: we denote the value $p^i \odot Z_5^i$ by s^i .

The output of the i -th round is $(Y_1^i \oplus t^i, Y_3^i \oplus t^i, Y_2^i \oplus u^i, Y_4^i \oplus u^i)$. In the last round (round 9) the MA layer is removed. Thus, the ciphertext is $(Y_1^9 || Y_2^9 || Y_3^9 || Y_4^9)$. The structure of a single round of IDEA is shown in Figure 1.

IDEA's key schedule is linear: each subkey is composed of bits selected from the key. However, the exact structure of the key schedule is crucial for our attacks and hence the entire key schedule is described in Table 2.

Table 2. The Key Schedule Algorithm of IDEA

Round	Z_1^i	Z_2^i	Z_3^i	Z_4^i	Z_5^i	Z_6^i
$i = 1$	0–15	16–31	32–47	48–63	64–79	80–95
$i = 2$	96–111	112–127	25–40	41–56	57–72	73–88
$i = 3$	89–104	105–120	121–8	9–24	50–65	66–81
$i = 4$	82–97	98–113	114–1	2–17	18–33	34–49
$i = 5$	75–90	91–106	107–122	123–10	11–26	27–42
$i = 6$	43–58	59–74	100–115	116–3	4–19	20–35
$i = 7$	36–51	52–67	68–83	84–99	125–12	13–28
$i = 8$	29–44	45–60	61–76	77–92	93–108	109–124
$i = 9$	22–37	38–53	54–69	70–85		

3 A New Non-trivial Relation Between the Three Operations of IDEA

In this section we present the new non-trivial relation between the three different operations of IDEA. The relation we present is a property of the MA layer. Since the property is independent of the round number, in this section we omit the round index in all the notations. The property is related to the XOR difference between the values in two encryptions. We denote the difference in the word X by ΔX .

Observation 1. *Assume that the XOR difference between the two intermediate encryption values in the input to the MA layer is of the form $(\Delta p, \Delta q) = (0, \alpha)$ for some α . Assume also that there is no key difference in the key word Z_5 (but there is no assumption whether there is a key difference in the subkey word Z_6). Then:*

1. *The least significant bit of the value $\Delta u \oplus \Delta t$ equals zero.*
2. *The average probability of the event $(\Delta u, \Delta t) = (8000_x, 8000_x)$ over all the possible keys is 2^{-16} (if $\alpha \neq 0$ or if there is a key difference in Z_6).*
3. *If α is non-zero or if there is a difference in Z_6 , then $\Sigma_{\nu, \tau} \Pr^2[(\Delta u, \Delta t) = (\nu, \tau)] = 2^{-23.72}$.*

We note that the first part of the observation is similar to observations that were used in [12, 16, 24].

If the MA layer was truly random, then the probability of the event $(\Delta u, \Delta t) = (8000_x, 8000_x)$ would be 2^{-32} . Hence, we have a differential with a much higher probability than expected.

The third part of the observation gives a much higher value than the corresponding value for a random function (which is 2^{-32}). The value discussed in the third part of the observation affects boomerang and rectangle attacks.

We shall now provide the proof of the observation: The proof uses the additive difference (module 2^{16}) between the two inputs, which we denote by δX . As there is no XOR difference in the first input word to the MA function ($\Delta p = 0$), then there is no additive difference as well, i.e., $\delta p = 0$. As there is no additive

difference in the subkey Z_5 , then $\Delta s = \delta s = 0$ as well. As $u = t \boxplus s$ then $\delta u = \delta t \boxplus \delta s = \delta t$. We use this relation in the proof:

1. $LSB(\Delta u) = LSB(\delta u) = LSB(\delta t) = LSB(\Delta t)$, where $LSB(w)$ denotes the least significant bit of the word w . Thus, $LSB(\Delta u \oplus \Delta t) = LSB(\Delta u) \oplus LSB(\Delta t) = 0$.
2. Since no assumption on α or the subkey difference in Z_6 was used (aside the fact that there is such a difference), we can assume that the value δt is randomly distributed. Hence, with probability 2^{-16} the difference is $\delta t = 8000_x$. In this case, $\delta u = 8000_x$ as well. However, $\delta t = 8000_x$ is equivalent to $\Delta t = 8000_x$. Thus, the probability of the event $(\Delta u, \Delta t) = (8000_x, 8000_x)$ is indeed 2^{-16} , as asserted.
3. We can write

$$\Sigma_{\nu,\tau} \Pr^2[(\Delta u, \Delta t) = (\nu, \tau)] = \Sigma_{\nu,\tau} (\Sigma_{\delta} \Pr[(\Delta u, \Delta t) = (\nu, \tau) \wedge (\delta t = \delta)])^2 = 2^{-32} \cdot \Sigma_{\nu,\tau} (\Sigma_{\delta} \Pr[(\Delta u, \Delta t) = (\nu, \tau)](\delta t = \delta))^2$$

where the last equality follows from the assumption that $\Pr[\delta t = \delta] = 2^{-16}$ for every δ . We calculated the last value explicitly by a computer program and got the value $\Sigma_{\beta,\gamma} \Pr^2[(\Delta u, \Delta t) = (\beta, \gamma)] = 2^{-23.72}$, as asserted.

Q.E.D.

4 A New Attack on 5-Round IDEA

In this section we present new attacks on 2.5-round, 3-round and 5-round IDEA based on the first relation established in Section 3.

We start with an observation due to Biryukov (according to [24]) and Demirci [12]. Let us examine the second and the third words in all the intermediate stages of the encryption. There is a relation between the values of these words and the outputs of the MA layer in the intermediate rounds that uses only XOR and modular addition, but not multiplication. Let $P = (P_1, P_2, P_3, P_4)$ be a plaintext and let $C = (C_1, C_2, C_3, C_4)$ be its corresponding ciphertext, then

$$\begin{aligned} &((((((((((((((((((P_2 \boxplus Z_2^1) \oplus u^1) \boxplus Z_3^2) \oplus t^2) \boxplus Z_2^3) \oplus u^3) \boxplus Z_3^4) \oplus t^4) \boxplus Z_2^5) \oplus u^5) \\ &\boxplus Z_3^6) \oplus t^6) \boxplus Z_2^7) \oplus u^7) \boxplus Z_3^8) \oplus t^8) \boxplus Z_2^9) = C_2. \end{aligned} \tag{1}$$

Similarly,

$$\begin{aligned} &((((((((((((((((((P_3 \boxplus Z_3^1) \oplus t^1) \boxplus Z_2^2) \oplus u^2) \boxplus Z_3^3) \oplus t^3) \boxplus Z_2^4) \oplus u^4) \boxplus Z_3^5) \oplus t^5) \\ &\boxplus Z_2^6) \oplus u^6) \boxplus Z_3^7) \oplus t^7) \boxplus Z_2^8) \oplus u^8) \boxplus Z_3^9) = C_3. \end{aligned} \tag{2}$$

Now, if we are interested only in the value of the least significant bit (LSB) of the words, modular addition is equivalent to XOR and we can simplify the above equations into:

$$\begin{aligned} &LSB(P_2 \oplus Z_2^1 \oplus u^1 \oplus Z_3^2 \oplus t^2 \oplus Z_2^3 \oplus u^3 \oplus Z_3^4 \oplus t^4 \oplus Z_2^5 \oplus u^5 \oplus Z_3^6 \oplus t^6 \oplus Z_2^7 \\ &\oplus u^7 \oplus Z_3^8 \oplus t^8 \oplus Z_2^9) = LSB(C_2), \end{aligned} \tag{3}$$

and

$$LSB(P_3 \oplus Z_3^1 \oplus t^1 \oplus Z_2^2 \oplus u^2 \oplus Z_3^3 \oplus t^3 \oplus Z_2^4 \oplus u^4 \oplus Z_3^5 \oplus t^5 \oplus Z_2^6 \oplus u^6 \oplus Z_3^7 \oplus t^7 \oplus Z_2^8 \oplus u^8 \oplus Z_3^9) = LSB(C_3). \quad (4)$$

Since $u^i = t^i \boxplus s^i$ then $LSB(u^i) = LSB(t^i \boxplus s^i)$, thus, $LSB(u^i \boxplus t^i) = LSB(s^i)$. Taking this into consideration and XORing the two above equations we obtain

$$LSB(P_2 \oplus P_3 \oplus Z_2^1 \oplus Z_3^1 \oplus s^1 \oplus Z_2^2 \oplus Z_3^2 \oplus s^2 \oplus Z_2^3 \oplus Z_3^3 \oplus s^3 \oplus Z_2^4 \oplus Z_3^4 \oplus s^4 \oplus Z_2^5 \oplus Z_3^5 \oplus s^5 \oplus Z_2^6 \oplus Z_3^6 \oplus s^6 \oplus Z_2^7 \oplus Z_3^7 \oplus s^7 \oplus Z_2^8 \oplus Z_3^8 \oplus s^8 \oplus Z_2^9 \oplus Z_3^9) = LSB(C_2 \oplus C_3). \quad (5)$$

This equation is called in [16] “the Biryukov-Demirci relation”.

Consider two plaintexts P^1 and P^2 . Denote the XOR difference between the encryptions of P^1 and P^2 (under the same secret key) in an intermediate value X by ΔX . Then, the XOR the equations given by P^1 and P^2 gives

$$LSB(P_2^1 \oplus P_3^1 \oplus P_2^2 \oplus P_3^2 \oplus \Delta s^1 \oplus \Delta s^2 \oplus \Delta s^3 \oplus \Delta s^4 \oplus \Delta s^5 \oplus \Delta s^6 \oplus \Delta s^7 \oplus \Delta s^8) = LSB(C_2^1 \oplus C_3^1 \oplus C_2^2 \oplus C_3^2). \quad (6)$$

Equation (6) is the basic equation used in all our attacks in this section.

4.1 A Distinguishing Attack on 2.5-Round IDEA

Consider a 2.5-round variant of IDEA of the form $KA \circ MA \circ KA \circ MA \circ KA$. For sake of simplicity we assume that the attack is on the first 2.5 rounds of IDEA, but the same attack holds for any 2.5 consecutive rounds of this form.

For a 2.5-round IDEA, Equation (6) is reduced to

$$LSB(P_2^1 \oplus P_3^1 \oplus P_2^2 \oplus P_3^2 \oplus \Delta s^1 \oplus \Delta s^2) = LSB(C_2^1 \oplus C_3^1 \oplus C_2^2 \oplus C_3^2). \quad (7)$$

Note that by the first part of the observation in Section 3, if the input XOR difference to the MA layer is of the form $(\Delta p, \Delta q) = (0, \alpha)$ then $\Delta s = 0$. In order to use this property, we consider pairs of plaintexts (P^1, P^2) such that $\Delta(X_1^1, X_2^1, X_3^1, X_4^1) = (0, \beta, 0, \gamma)$ for arbitrary values of β and γ . For these pairs $\Delta Y_1^1 = \Delta Y_3^1 = 0$ (independent of the values Z_1^1, Z_3^1), and hence $\Delta p^1 = 0$. Therefore, the required property holds and $\Delta s^1 = 0$. We note that the same idea was used (to some extent) in [16].

Similarly, if we take only ciphertext pairs satisfying $\Delta(Y_1^3, Y_2^3, Y_3^3, Y_4^3) = (0, 0, \beta', \gamma')$ for arbitrary values of β' and γ' , then $(\Delta p^2, \Delta q^2) = (0, \alpha')$ for some α' , and hence $\Delta s^2 = 0$.

If the plaintext/ciphertext pair $((P^1, C^1), (P^2, C^2))$ satisfies both differential relations required above, Equation (7) is further reduced into

$$LSB(P_2^1 \oplus P_3^1 \oplus P_2^2 \oplus P_3^2) = LSB(C_2^1 \oplus C_3^1 \oplus C_2^2 \oplus C_3^2). \quad (8)$$

This is a simple linear relation that can be checked easily since only bits of the plaintexts and the ciphertexts are involved in the equation.

Based on these observations, we can mount a simple distinguishing attack on 2.5-round IDEA, using the following algorithm:

1. Ask for the encryption of 2^{18} plaintexts of the form (A, Z, B, W) , where A and B are fixed and Z and W assume arbitrary random values.
2. Insert the ciphertexts into a hash table sorted by the first two words.
3. For every pair of ciphertexts in the same bin of the hash table, check whether Equation (8) holds for the corresponding plaintext/ciphertext pair.
4. If there is a pair for which the equation does not hold, conclude that the cipher is not 2.5-round IDEA. If there is no such pair, conclude that the cipher is 2.5-round IDEA.

Due to the structure of the plaintexts, for every pair of plaintexts the first differential requirement holds. For every pair of ciphertexts in the same bin of the hash table, the second requirement also holds. Hence, for all the checked pairs Equation (8) should be satisfied for 2.5-round IDEA.

The 2^{18} plaintexts can be combined into about 2^{35} possible pairs, and a fraction of 2^{-32} of them is expected to have ciphertext difference of the form $(0, 0, \beta', \gamma')$. Hence, the expected number of pairs analyzed in Step 3 is eight. If there is a pair for which the equation does not hold, we know for sure that the cipher is not 2.5-round IDEA. On the other hand, for a random permutation, the probability that the equation holds for all the eight pairs is $1/256$. Hence, the distinguisher succeeds with probability greater than 99.5%.

Since the second and the third steps of the attack are implemented using a hash table, the time complexity of the attack is dominated by the time complexity of the encryptions in the first step of the attack. Hence, the data complexity of the attack is 2^{18} chosen plaintexts and the time complexity is 2^{18} encryptions.

4.2 A Key Recovery Attack on 3-Round IDEA

The 2.5-round distinguisher can be extended to an attack on 3-round IDEA of the form $E = KA \circ MA \circ KA \circ MA \circ KA \circ MA$ by guessing the subkey of the last MA layer and applying the distinguishing attack to the first 2.5 rounds. In this case, the data complexity is slightly increased, since more pairs are required in the last step of the attack in order to discard all the wrong key values.

The attack algorithm is the following:

1. Ask for the encryption of 2^{19} plaintexts of the form (A, Z, B, W) , where A and B are fixed and Z and W assume arbitrary random values.
2. For every guess of the 32-bit subkey of the last MA layer:
 - (a) Partially decrypt all the ciphertexts through the last MA layer and insert the resulting Y^3 values into a hash table sorted by the first 32 bits.
 - (b) For every pair of values in the same bin of the hash table, check whether Equation (8) holds for the corresponding plaintext/ciphertext pair.
 - (c) If there is a pair for which the equation does not hold, discard the subkey guess. Otherwise, keep the subkey guess.
3. Output all the subkey guesses that were not discarded.

Since there are 2^{19} plaintexts, then there are about 2^{37} possible pairs, and about 32 pairs are examined in Step 2(b). Hence, for a wrong key guess the

probability that the equation holds for all the pairs is 2^{-32} . Therefore, only few possible key guesses remain, including the right key. The filtering can be further improved by enlarging the data structure by a small factor.

The time complexity of the attack is dominated by Step 2(b) which contains decrypting all ciphertexts under all the subkey guesses. The data complexity of the attack is 2^{19} chosen plaintexts and the time complexity of the attack is equivalent to $2^{19} \times 2^{32} \times (1/6) \approx 2^{48.5}$ 3-round encryptions. Note that the attack recovers only 32 bits of the master key and the rest of the key has to be found using other techniques.

We note that a similar attack can be mounted on a 3-round variant of IDEA of the form $E = MA \circ KA \circ MA \circ KA \circ MA \circ KA$. The only difference is that in this case the attack is performed in the decryption direction. The time and data complexities remain unchanged.

The two extensions can be combined to an attack on a 3.5-round variant of IDEA of the form $E = MA \circ KA \circ MA \circ KA \circ MA \circ KA \circ MA$. However, in this case the data and time complexities are worse than the complexities of the best known attack on 3.5-round IDEA. This follows from the fact that while in the 3-round attacks we could guarantee that one of the differential conditions holds, in the 3.5-round attack this is not the case.

4.3 Attack on 5-Round IDEA

In this section we devise an attack on a 5-round variant of IDEA starting with the second half of round 3. Choosing round 3 as the starting point of the attack is the optimal round, as described later.

First, we consider a 4.5-round attack starting at the beginning of round 4. For this variant, the Equation (6) is transformed into

$$LSB(P_2^1 \oplus P_3^1 \oplus P_2^2 \oplus P_3^2 \oplus \Delta s^4 \oplus \Delta s^5 \oplus \Delta s^6 \oplus \Delta s^7) = LSB(C_2^1 \oplus C_3^1 \oplus C_2^2 \oplus C_3^2). \quad (9)$$

In our attack we use pairs of plaintexts with XOR difference $\Delta(X_1^4, X_2^4, X_3^4, X_4^4) = (0, \beta, 0, \gamma)$, thus, $\Delta s^4 = 0$. In order to calculate Δs^i for $5 \leq i \leq 7$, we guess part of the master key and partially decrypt the ciphertexts through the last three rounds.

In order to calculate the required Δs^i values, we guess the subkeys $Z_4^8, Z_3^8, Z_2^8, Z_1^8, Z_6^7, Z_5^7, Z_4^7, Z_3^7, Z_2^7, Z_1^7, Z_6^6, Z_5^6$ that allow to partially decrypt two rounds, and the subkeys Z_1^6, Z_2^6, Z_5^5 that allow to calculate the value Δs^5 . However, it appears that all these 15 subkeys use only 103 bits of the master key, whereas bits 100–124 of the master key remain unused. Hence, we can guess 103 bits of the master key, and for each guess we can check whether the equation holds for the plaintext/ciphertext pairs. We note that finding the right subkey requires about 128 pairs for the analysis, which can be constructed from about 16 chosen plaintexts. We also note that starting the attack in a different round would require guessing more subkey bits.

In order to extend the attack to 5 rounds, we guess the subkey of the MA layer in round 3. This does not increase the time complexity since the relevant subkey is composed of bits 50–81 of the master key that are included in the 103

bits we guess in the 4.5-round attack. However, this additional half round affects the data complexity of the attack.

The only remaining issue is getting pairs of plaintexts with difference $\Delta(X_1^4, X_2^4, X_3^4, X_4^4) = (0, \beta, 0, \gamma)$. Since for every guess of the *MA* layer of round 3 different plaintext pairs are needed to fulfill this differential requirement, this attack uses known plaintexts instead of chosen plaintexts. We start with 2^{19} known plaintexts that compose 2^{37} possible pairs. For each subkey guess of the *MA* layer of round 3, we partially encrypt all the plaintexts and choose the pairs that have difference $\Delta(X_1^4, X_2^4, X_3^4, X_4^4) = (0, \beta, 0, \gamma)$. We expect 32 such pairs, and these pairs are used in the sequel of the attack. The time complexity of this step is negligible compared to the time complexities of the other steps of the attack.

The attack algorithm is as follows:

1. Ask for the encryption of 2^{19} known plaintexts.
2. For each guess of key bits 50–81, perform the following:
 - (a) Partially encrypt the plaintexts through the *MA* layer of round 3 and insert the resulting X^4 values to a hash table indexed by the first and the third words.
 - (b) For each guess of key bits 0–49, 82–99,¹ and 125–127 and for all the colliding pairs, perform the following:
 - i. Partially decrypt all the pairs through rounds 7 and 6, and the *MA* layer of round 5.
 - ii. Verify that Equation (9) holds for all of the pairs. If no, discard the key guess.
 - (c) If the key guess passed the filtering, perform exhaustive search on the remaining 25 key bits.

As we mentioned before, for every guess of key bits 50–81, we expect that 32 pairs are analyzed in Step 2(b) of the attack. Hence, the probability that a wrong key guess passes the filtering is 2^{-32} . Thus, we expect that about $2^{103} \cdot 2^{-32} = 2^{71}$ key guesses enter Step 2(c). Thus, the time complexity of Step 2(c) is expected to be equivalent to $2^{25} \cdot 2^{71} = 2^{96}$ encryptions in total.

Therefore, the time complexity of the attack is dominated by the partial decryptions of Step 2(b). We observe that this step can be optimized. Note that half of the key guesses are discarded after the first pair, half of the remaining key guesses are discarded after the second pair, etc. Hence, instead of decrypting all the pairs at once, the attacker can decrypt the first pair and check whether the equation holds, then (if the key guess was not discarded) decrypt the second pair and check the equation for it, etc. Using this improvement, the time complexity of this step is $2^{103} + 2^{102} + 2^{101} + \dots \approx 2^{104}$ partial decryptions, which are roughly equivalent to 2^{103} full encryptions.

Hence, the data complexity of the attack is 2^{19} known plaintexts and the time complexity is 2^{103} encryptions.

¹ Note that key bits 50–81 are already guessed.

5 Related-Key Attack on 7.5-Round IDEA

In this section we present a related-key attack on the first 7.5 rounds of IDEA. The 7.5-round related-key attack uses similar relations as the 5-round known plaintext attack. In the attack we use the difference between the keys to construct pairs of plaintexts for which the intermediate values (when encrypted under the two different keys) are equal for 2.5 rounds. For such pairs of plaintexts, Equation (6) is reduced to a much simpler one.

Let the K and K^* be two keys such that they are equal in all bits but bit 34 and any non-empty subset of bits $\{41, 42, \dots, 49\}$. Let P and P^* be the two plaintexts, such that Y^2 and Y^{2*} , the corresponding intermediate encryption values after the KA layer of round 2, satisfy:

$$Y_1^2 = Y_1^{2*}; \quad Y_2^2 = Y_2^{2*}; \quad Y_3^2 = Y_3^{2*}; \quad Y_4^2 = Y_4^{2*} \quad (10)$$

In such pair, the intermediate encryption values are equal until the MA layer of round 4. In that MA layer, the input difference is $(\Delta p^4, \Delta q^4) = (0, 0)$ and the key difference affects only Z_6^4 . Hence, by the observation presented in Section 3, $\Delta s^2 = \Delta s^3 = \Delta s^4 = 0$.

Therefore, for such pair Equation (6) is reduced to

$$LSB(P_2 \oplus P_3 \oplus P_2^* \oplus P_3^* \oplus \Delta s^1 \oplus \Delta s^5 \oplus \Delta s^6 \oplus \Delta s^7) = LSB(C_2 \oplus C_3 \oplus C_2^* \oplus C_3^*). \quad (11)$$

Hence, if the attacker is able to construct plaintext pairs satisfying Equation (10), he can partially encrypt/decrypt the plaintext/ciphertext pairs through rounds 1, 7, 6, and 5 and check whether Equation (11) is satisfied. In order to do so, the attacker has to guess the subkeys Z_1^1, Z_3^1, Z_5^1 for the partial encryption and $Z_5^5, Z_1^6, Z_2^6, Z_6^6, Z_6^6, Z_1^7, Z_6^7, Z_1^8, Z_4^8$ for the partial decryption. However, these 18 subkeys use only 103 bits of the master key, and hence guessing these key bits and checking whether Equation (11) holds for some plaintext/ciphertext pairs satisfying Equation (10) yields an attack faster than exhaustive key search.

Constructing pairs of plaintexts satisfying Equation (10) is not a trivial operation. However, if we use the known plaintext model and take sufficiently many plaintexts, then Equation (10) may be satisfied sufficiently many times. A naive approach would be to partially encrypt all the given known plaintexts through round 1 and the KA layer of round 2, and to find the relevant pairs. However, even in an optimized manner, this approach would result in guessing 96 key bits, which combined with the known plaintext nature of the attack results in a time complexity of least 2^{128} 1-round IDEA encryptions.

Therefore, we use a modified approach. We use $2^{42.5}$ known plaintexts encrypted under two related keys (a total of $2^{43.5}$ related-key known plaintexts), and partially encrypt them through the KA layer of round 1. After the KA layer, we consider only the pairs that have difference $(0, 0040_x, 0, 0040_x)$. Such pairs have difference $(0, 0, 0040_x, 0040_x)$ at the input to the KA layer of round 2, independent of the value of the subkeys Z_5^1, Z_6^1 . With probability 1/2 the difference in the third word is canceled by the key difference, and with probability 2^{-16} the difference in the fourth word is canceled by the key difference, leading

to a pair that satisfies Equation (10). Hence, the required pairs are detected in a two steps algorithm. First the attacker guesses the values of the subkeys Z_1^1, Z_2^1, Z_3^1 , and Z_4^1 and finds the pairs having difference $(0, 0040_x, 0, 0040_x)$ after the first KA layer. Most of the pairs are filtered at this stage. Then the attacker further guesses the values of the subkeys Z_5^1, Z_6^1, Z_3^2 , and Z_4^2 and checks which of the remaining pairs satisfy Equation (10).

The attack algorithm on 7.5-round IDEA is as follows:

1. Ask for $2^{42.5}$ known plaintexts encrypted under K and denote the set of plaintexts and ciphertexts by $SetP$.
2. Ask for $2^{42.5}$ known plaintexts encrypted under K^* and denote the set of plaintexts and ciphertexts by $SetP^*$.
3. For each guess of the subkeys Z_1^1, Z_2^1, Z_3^1 , and Z_4^1 :
 - (a) Partially encrypt all plaintexts in $SetP$ and in $SetP^*$ through the KA layer of round 1.
 - (b) Find all pairs of Y^1 (encrypted under K) and Y^{1*} (encrypted under K^*) such that $Y^1 \oplus Y^{1*} = (0, 0040_x, 0, 0040_x)$.
 - (c) For each such pair, and each guess of Z_5^1, Z_6^1, Z_3^2 , and Z_4^2 :
 - i. If the pair satisfies Equation (10), guess $Z_5^5, Z_1^6, Z_2^6, Z_5^6, Z_6^6, Z_1^7 - Z_6^7$, and $Z_1^8 - Z_4^8$ and verify whether Equation (11) is satisfied.
 - ii. If the equation is not satisfied — discard the subkey guess.
4. For each remaining subkey, exhaustively try all 25 remaining subkey bits, and output the remaining key.

There are 2^{85} pairs of plaintexts, of which $2^{85} \cdot 2^{-64} = 2^{21}$ have difference $(0, 0040_x, 0, 0040_x)$ after the KA layer of round 1. For each guess of Z_5^1, Z_6^1, Z_3^2 , and Z_4^2 , about $2^{21} \cdot 2^{-17} = 16$ pairs have a zero difference after the KA layer of round 2, satisfying Equation (10). For a correct subkey guess, all these pairs should satisfy Equation (11). For wrong subkey guesses, the probability that Equation (11) is satisfied for all the pairs is 2^{-16} . There are 2^{103} possible subkeys, and hence the number of subkeys that enter Step 4 is expected to be $2^{103} \cdot 2^{-16} = 2^{87}$.

The time complexity of the attack is thus dominated by Step 3 (Steps 1 and 2 have time complexity of $2^{42.5}$ encryptions each, and Step 4 has time complexity of $2^{87} \cdot 2^{25} = 2^{112}$ trial encryptions). Step 3(a) is repeated 2^{64} times, and each time $2^{43.5}$ values are partially encrypted through one KA layer. Hence, the time complexity of this step is $2^{64} \cdot 2^{43.5} = 2^{107.5}$ partial encryptions. Step 3(b) can be executed efficiently using a hash table. In Step 3(c)(i) only 2^{21} pairs (or 2^{22} values) are analyzed but this step requires guessing 32 more bits (Z_3^2 and Z_4^2 are covered by the bits guessed in Step 3(a)). Thus, the time complexity of the first part of this step (finding the pairs satisfying Equation (10)) is $2^{64} \cdot 2^{22} \cdot 2^{32} = 2^{118}$ 1-round decryptions. The time complexity of the second part of Step 3(c)(i) (checking whether Equation (11) is satisfied) is much lower, as even though 9 more key bits are guessed, there are only 32 pairs (or 64 values) that enter this step. Thus, the total time complexity of the attack is about $2^{118} \cdot \frac{1}{7.5} = 2^{115.1}$ 7.5-round IDEA encryptions.

6 Summary and Conclusions

In this paper we presented several new results on the block cipher IDEA: The first non-trivial relation involving all the three different operations of IDEA, a known-plaintext 5-round attack, a related-key attack on 7.5-round IDEA (with two keys) and a related-key rectangle attack on 7-round IDEA (with four keys). These results are by far the best known attacks against reduced-round variants of the cipher.

Our paper shows that the linear key schedule of IDEA makes the cipher relatively vulnerable to attacks that guess vast amounts of the key. However, despite our findings, the full IDEA still resists all known attacks.

References

- [1] Eli Biham, Alex Biryukov, Adi Shamir, *Miss in the Middle Attacks on IDEA and Khufu*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 124–138, Springer-Verlag, 1999.
- [2] Eli Biham, Orr Dunkelman, Nathan Keller, *The Rectangle Attack – Rectangling the Serpent*, Advances in Cryptology, proceedings of EUROCRYPT '01, Lecture Notes in Computer Science 2045, pp. 340–357, Springer-Verlag, 2001.
- [3] Eli Biham, Orr Dunkelman, Nathan Keller, *New Results on Boomerang and Rectangle Attacks*, proceedings of Fast Software Encryption 9, Lecture Notes in Computer Science 2365, pp. 1–16, Springer-Verlag, 2002.
- [4] Eli Biham, Orr Dunkelman, Nathan Keller, *New Combined Attacks on Block Ciphers*, proceedings of Fast Software Encryption 12, Lecture Notes in Computer Science 3557, pp. 126–144, Springer-Verlag, 2005.
- [5] Eli Biham, Orr Dunkelman, Nathan Keller, *Related-Key Boomerang and Rectangle Attacks*, Advances in Cryptology, proceedings of EUROCRYPT'05, Lecture Notes in Computer Science 3494, pp. 507–525, Springer-Verlag, 2005.
- [6] Alex Biryukov, Jorge Nakahara Jr., Bart Preneel, Joos Vandewalle, *New Weak-Key Classes of IDEA*, proceedings of Information and Communications Security 4, Lecture Notes in Computer Science 2513, pp. 315–326, Springer-Verlag, 2002.
- [7] Nikita Borisov, Monica Chew, Robert Johnson, David Wagner, *Multiplicative Differentials*, proceedings of Fast Software Encryption 9, Lecture Notes in Computer Science 2365, pp. 17–33, Springer-Verlag, 2002.
- [8] Johan Borst, Lars R. Knudsen, Vincent Rijmen, *Two Attacks on Reduced Round IDEA*, Advances in Cryptology, proceedings of EUROCRYPT '97, Lecture Notes in Computer Science 1233, pp. 1–13, Springer-Verlag, 1997.
- [9] Joan Daemen, René Govaerts, Joos Vandewalle, *Cryptanalysis of 2.5 Rounds of IDEA (Extended Abstract)*, technical report 93/1, Department of Electrical Engineering, ESAT–COSIC, Belgium, 1993.
- [10] Joan Daemen, René Govaerts, Joos Vandewalle, *Weak Keys for IDEA*, Advances in Cryptology, proceedings of CRYPTO '93, Lecture Notes in Computer Science 773, pp. 224–231, Springer-Verlag, 1994.
- [11] Hüseyin Demirci, *Square-like Attacks on Reduced Rounds of IDEA*, proceedings of Selected Areas in Cryptography 2002, Lecture Notes in Computer Science 2595, pp. 147–159, Springer-Verlag, 2003.

- [12] Hüseyin Demirci, Ali A. Selçuk, Erkan Türe, *A New Meet-in-the-Middle Attack on the IDEA Block Cipher*, proceedings of Selected Areas in Cryptography 2003, Lecture Notes in Computer Science 3006, pp. 117–129, Springer-Verlag, 2004.
- [13] Philip Hawkes, *Differential-Linear Weak Keys Classes of IDEA*, Advances in Cryptology, proceedings of EUROCRYPT '98, Lecture Notes in Computer Science 1403, pp. 112–126, Springer-Verlag, 1998.
- [14] P. Hawkes, L. O'Connor, *On Applying Linear Cryptanalysis to IDEA*, Advances in Cryptology - Proceedings of ASIACRYPT'96, Lecture Notes in Computer Science 1163, pp. 105–115, Springer-Verlag, 1996.
- [15] Seokhie Hong, Jongsung Kim, Guil Kim, Sangjin Lee, Bart Preneel, *Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192*, proceedings of Fast Software Encryption 12, Lecture Notes in Computer Science 3557, pp. 368–383, Springer-Verlag, 2005.
- [16] Pascal Junod, *New Attacks Against Reduced-Round Versions of IDEA*, proceedings of Fast Software Encryption 12, Lecture Notes in Computer Science 3557, pp. 384–397, Springer-Verlag, 2005.
- [17] John Kelsey, Bruce Schneier, David Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, Advances in Cryptology, proceedings of CRYPTO '96, Lecture Notes in Computer Science 1109, pp. 237–251, Springer-Verlag, 1996.
- [18] John Kelsey, Tadayoshi Kohno, Bruce Schneier, *Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent*, proceedings of Fast Software Encryption 7, Lecture Notes in Computer Science 1978, pp. 75–93, Springer-Verlag, 2000.
- [19] Jongsung Kim, Guil Kim, Seokhie Hong, Sangjin Lee, Dowon Hong, *The Related-Key Rectangle Attack — Application to SHACAL-1*, proceedings of ACISP 2004, Lecture Notes in Computer Science 3108, pp. 123–136, Springer-Verlag, 2004.
- [20] Xuejia Lai, James L. Massey, Sean Murphy, *Markov Ciphers and Differential Cryptanalysis*, Advances in Cryptology - Proceedings of EUROCRYPT'91, Lecture Notes in Computer Science 547, pp. 17–38, Springer-Verlag, 1992.
- [21] Willi Meier, *On the Security of the IDEA Block Cipher*, Advances in Cryptology, proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765, pp. 371–385, Springer-Verlag, 1994.
- [22] Jorge Nakahara Jr., Paulo S.L.M. Barreto, Bart Preneel, Joos Vandewalle, Hae Y. Kim, *SQUARE Attacks Against Reduced-Round PES and IDEA Block Ciphers*, IACR Cryptology ePrint Archive, Report 2001/068, 2001.
- [23] Jorge Nakahara Jr., Vincent Rijmen, Bart Preneel, Joos Vandewalle, *The MESH Block Ciphers*, proceedings of Information Security Applications, 4th International Workshop, WISA 2003, Lecture Notes in Computer Science 2908, pp. 458–473, Springer-Verlag, 2004.
- [24] Jorge Nakahara Jr., Bart Preneel, Joos Vandewalle, *The Biryukov-Demirci Attack on Reduced-Round Versions of IDEA and MESH Ciphers*, proceedings of ACISP 2004, Lecture Notes in Computer Science 3108, pp. 98–109, Springer-Verlag, 2004.
- [25] NESSIE, *Performance of Optimized Implementations of the NESSIE Primitives*, NES/DOC/TEC/WP6/D21/a, available on-line at <http://www.nessie.eu.org/nessie>.
- [26] Havard Raddum, *Cryptanalysis of IDEA-X/2*, proceedings of Fast Software Encryption 10, Lecture Notes in Computer Science 2887, pp. 1–8, Springer-Verlag, 2003.
- [27] David Wagner, *The Boomerang Attack*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 156–170, 1999.

A A Related-Key Rectangle Attack on 7-Round IDEA

In this appendix we use the third part of the observation in Section 3 to improve the 6.5-round related-key rectangle attack presented in [5] and to devise a related-key rectangle attack on 7-round IDEA. Due to space constraints, we present only the main idea of the attacks and the final results. The detailed description of the attacks appears in the full version of the paper.

We start by devising a new related-key boomerang distinguisher for 5.5-round IDEA. The data complexity of the distinguisher is worse than that of the distinguisher used in [5], but it can be used to devise better key recovery attacks. We note that the distinguisher used in [5] can be also improved using similar techniques. This improvement is also described in the full version of the paper.

The new 5.5-round distinguisher is applicable for rounds 1.5–6. The first related-key differential starts after the KA layer of round 1 with the difference $(0, 0040_x, 0, 0040_x)$ and ends after the MA layer of round 4. The key difference is in bit 34, and any non-empty subset of bits $\{41, 42, \dots, 49\}$. The second related-key differential starts at the beginning of round 5 with the difference $(0, 8000_x, 0, 0)$ and key difference in key bit 91. This difference evolves into a zero difference after the MA layer of round 6 with probability 1.

The second differential is quite standard. It is based on cancelling the difference in the second word using the key difference in bit 91 (i.e., $\Delta K_1 = e_{91}$). Then, the zero difference is preserved until key bit 91 is used again in the subkey Z_4^7 .

The first differential is a bit more complicated. A pair with input difference $\alpha = (0, 0040_x, 0, 0040_x)$ to the MA layer of round 1 has difference $(0, 0, 0040_x, 0040_x)$ after the MA layer with probability 1. With probability $1/2$ the key difference cancels the data difference in the third word, and with probability 2^{-16} the key difference cancels the data difference in the fourth word. Thus, with probability 2^{-17} , the pair has a zero difference after the KA layer of round 2. This zero difference is preserved until the last multiplication in the MA layer of round 4. Hence, in that MA layer both Δp^4 and the key difference in Z_5^4 are zero. Thus, we can apply the third part of the observation in Section 3 to obtain $\hat{p} = 2^{-17} \cdot 2^{-11.86} = 2^{-28.86}$. The key difference ΔK_0 can be any of 511 possible values. We use the value $\Delta K_0 = e_{34,49}$, but it can be any of the other values without affecting our attack.

Using these differentials, we get a 5.5-round related-key boomerang distinguisher that uses $2^{59.32}$ adaptive chosen plaintexts and ciphertexts ($2^{57.32}$ values are encrypted/decrypted using four different keys).

We now present a related-key rectangle attack [5, 15, 19] on the first 6.5 rounds of IDEA based on the distinguisher presented above. The attack algorithm mostly follows the attack algorithm presented in [3] with the few modifications needed due to the related-key nature of the attack.

Let K_a, K_b, K_c, K_d be the related keys such that $K_b = K_a \oplus \Delta K_0$, $K_c = K_a \oplus \Delta K_1$, and $K_d = K_c \oplus \Delta K_0$. The attack algorithm is as follows:

1. Data Collection Phase

- (a) Generate 2^{35} structures $S_1^a, \dots, S_{2^{35}}^a$ of 2^{28} plaintexts each, where in each structure the first word, the six least significant bits of the second word,

- and the 14 least significant bits of the third word are fixed. Ask for the encryption of the structures under K_a .
- (b) Flip bit 6 of the second word and bit 13 of the third word of any plaintext encrypted under K_a , and ask for the encryption of the resulting plaintexts under K_b (to obtain $S_1^b, \dots, S_{2^{35}}^b$).
 - (c) Generate 2^{35} structures $S_1^c, \dots, S_{2^{35}}^c$ of 2^{28} plaintexts each, where in each structure the first word, the six least significant bits of the second word, and the 14 least significant bits of the third word are fixed. Ask for the encryption of the structures under K_c .
 - (d) Flip bit 6 of the second word and bit 13 of the third word of any plaintext encrypted under K_c , and ask for the encryption of the resulting plaintexts under K_d (to obtain $S_1^d, \dots, S_{2^{39}}^d$).
- 2. Finding Candidate Quartets**
- (a) Find all pairs of ciphertexts $C_a \in S_i^a$ and $C_c \in S_j^c$, such that they have the same value in the first, the second, and the third words.
 - (b) For each such pair, check whether there are pairs of ciphertexts $C_b \in S_i^b$ and $C_d \in S_j^d$, such that they have the same value in the first, the second, and the third words. If such a pair exists — transfer (P_a, P_b, P_c, P_d) , the corresponding plaintexts, to analysis.
- 3. Analysis of Candidate Quartets**
- (a) Initialize 2^{64} counters, each corresponds to a different guess of $Z_1^2, Z_1^3, Z_1^4, Z_7^4$.
 - (b) For each subkey guess of $Z_1^2, Z_1^3, Z_1^4, Z_7^4$ and each candidate quartet, check whether the partial encryption and partial decryption of the pairs of the quartet lead to the required differences. If this is the case increment the respective counter.
- 4. Output:** Output all subkey guesses whose counter has values greater than 8.

The analysis presented in the full version of the paper shows that the data complexity of the attack is 2^{65} related-key chosen plaintexts and the time complexity is 2^{87} memory accesses.

The 6.5-round attack can be extended to an attack on rounds 1–7 of IDEA by partially decrypting all the ciphertexts under all possible values of the key of the last MA layer, and applying the 6.5-round attack. A trivial implementation of this approach would lead to an attack that requires $2^{32} \cdot 2^{87} = 2^{119}$ memory accesses, and a data complexity of 2^{65} related-key chosen plaintexts.

However, we improve this result by observing that there are 12 shared bits between the subkeys Z_6^7 and Z_2^1 . This allows us to filter most of the wrong candidate quartets, by evaluating the difference after the addition in the KA layer of round 1. The improved attack is described in detail in the full version of the paper. The data complexity of the attack is 2^{65} related-key chosen plaintexts and the time complexity is 2^{111} memory accesses. Using the conversion of three clock cycles for one memory access, and the time measurements of the NESSIE project [25], these 2^{111} memory accesses are equivalent to $2^{104.2}$ 7-round IDEA encryptions.

New Approach for Selectively Convertible Undeniable Signature Schemes

Kaoru Kurosawa¹ and Tsuyoshi Takagi²

¹ Ibaraki University, Japan

kurosawa@mx.ibaraki.ac.jp

² Future University-Hakodate, Japan

takagi@fun.ac.jp

Abstract. In this paper, we propose a new approach for constructing selectively convertible undeniable signature schemes, and present two efficient schemes based on RSA. Our approach allows a more *direct* selective conversion than the previous schemes, and the security can be proved formally. Further, our disavowal protocols do not require parallelization techniques to reach a significant soundness probability. Also, our second scheme is the first selectively convertible scheme which is provably secure without random oracles.

Keywords: undeniable signature, selective conversion, RSA.

1 Introduction

1.1 Background

The concept of undeniable signature (**US**) schemes was introduced by Chaum and van Antwerpen [10]. In an US scheme, the signer issues an undeniable signature τ which is not publicly verifiable. She then proves the validity or invalidity of τ in zero-knowledge by running a confirmation protocol or disavowal protocol with the receiver. US schemes have found various applications in cryptography such as in licensing software [10], electronic cash [11,2,31], electronic voting and auctions. Then there have been a wide range of research covering a variety of different schemes for undeniable signatures over the past 15 years [7,1,9,8,19,14,18,25,4,17,16,22,3,26,27].

Recently, the security of Chaum's US scheme was proved formally in the random oracle model by [28]. Laguillaumie and Vergnaud showed an US scheme which is secure in the standard model under the strong Diffie-Hellman (DH) assumption [23]. The relations among the security notions for US schemes was given by [21].

The notion of *convertible* US schemes was introduced by Boyar et al. [1]. A *selectively* convertible US scheme allows the signer to convert an undeniable signature τ into a regular signature by releasing a piece of information α at a later time. *All* conversion means that the signer can convert all undeniable signatures into regular ones. They showed that if there exists a digital signature (**DS**) scheme, then there exists a convertible US scheme. However, this construction is not practical.

Damgård and Pedersen showed two selectively convertible US scheme schemes based on ElGamal signature scheme [14]. In their schemes, a part of the ElGamal signature is encrypted by Rabin encryption scheme or by ElGamal encryption scheme. However, invisibility is not proved in these schemes¹, where the invisibility means that we cannot decide if (m, τ) is a valid (message, undeniable signature) pair. Note that the invisibility is an essential property required for US schemes from the definition.

Gennaro-Krawczyk-Rabin proposed an RSA-based US scheme which allows all conversion efficiently [18].² They also showed a method of selective conversion such that the signer releases a non-interactive proof which shows that (m, τ) is a valid (message, undeniable signature) pair.

1.2 Our Contribution

In this paper, we propose a new approach for constructing selectively convertible undeniable signature schemes, and present two efficient schemes based on RSA. Our approach allows a more *direct* selective conversion than the previous schemes, and the security can be proved formally. Further, our disavowal protocols do not require parallelization techniques to reach a significant soundness probability. Also, our second scheme is the first selectively convertible US scheme whose security can be proved without random oracles.

A selectively convertible US scheme has two modes, the US signature issuing mode and the selective conversion mode. In our approach, we consider a DS signature issuing mode as well which is described as follows: For a message m ,

- The signer issues an undeniable signature τ in the US mode.
- In the DS mode, the signer issues σ as a regular signature on m .
- In the selective conversion mode, the signer releases σ (which is the same as above) to convert the already issued undeniable signature τ into a regular signature. By using σ , the validity of (m, τ) is made publicly verifiable.

We first formalize such US schemes as two-sided undeniable/signature schemes (“two-sided scheme” for short). In the security model, we consider adversaries who have access to both the DS-sign oracle and the US-sign oracle. Adversaries then try to forge a digital signature σ (DS-forgery) or an undeniable signature τ (US-forgery). See Figure 1. Both types of forgery must be impossible, and invisibility must be satisfied.

We next show an efficient two-sided scheme based on RSA signature scheme and Paillier’s encryption scheme [29]. In this scheme, the public-key is an RSA modulus $N(= pq)$.

¹ In Sec.5.1 and Sec.5.2 of [14], the authors wrote only that “We therefore conjecture that ...” on the invisibility of their schemes.

² GRK US scheme assumes that there exists an encoding method of messages so that the RSA-based DS scheme is unforgeable. However, no such encoding method is known in the standard model. Hence GRK US scheme is secure in the random oracle model only currently.

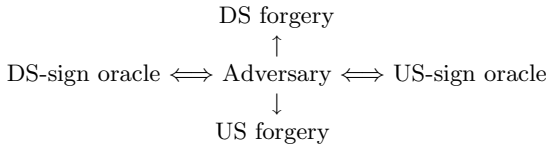


Fig. 1. Adversary in Two-sided scheme

- Our DS mode is the same as the RSA signature scheme with $e = N$. That is, the signer issues a digital signature $\sigma \in Z_N^*$ on a message m such that

$$\sigma^N = H(m) \pmod N,$$

where H is a hash function.

- Now replace $\text{mod}N$ with $\text{mod}N^2$ in the above equation. Then we obtain that

$$\sigma^N = H(m) + \tau N \pmod{N^2} \tag{1}$$

for some $\tau \in Z_N$. We consider that this τ is an undeniable signature on m . That is, in the US mode, the signer issues the above τ as an undeniable signature.

- In the selective conversion mode, the signer releases σ (which is the RSA signature on m) to convert the already issued τ into a regular signature. The validity of (m, τ) is publicly verified by checking eq.(1).

This piece of information σ released for selective conversion is smaller than that of GRK US scheme [18], where the latter is based on the Fiat-Shamir heuristic.³

Not only the above technique is new, but also our confirmation and disavowal protocols are based on a novel approach. In particular, our (zero-knowledge) disavowal protocol does not require parallelization techniques to reach a significant soundness probability. In the previous US schemes, only confirmation protocols are known which do not require parallelization techniques.

We then prove the security of our scheme in the random oracle model. Roughly speaking, our scheme relies on RSA assumption and the N th residuosity assumption.⁴

Finally, we show the first selectively convertible US scheme which is provably secure in the standard model. It is a two-sided scheme, and it is obtained by applying our technique to Cramer-Shoup DS scheme [13] which is known to be secure in the standard model.

Remark 1. In GRK US scheme [18], $N = pq$, where p and q must be safe primes. Galbraith et al. showed a method which can eliminate this restriction [17]. Our schemes are totally different from [18,17], and p and q can be any primes.

³ Since our scheme does not use the Fiat-Shamir heuristic, it uses one random oracle H while GRK scheme must use two random oracles (see footnote 2).

⁴ On the other hand, GRK US scheme [18] relies on RSA assumption and DDH assumption over Z_N^* , where the security model does not consider DS-sign oracle nor DS forgery.

2 Model and Definitions

For an algorithm A and its input x , we write $y \leftarrow A(x)$ if y is an output of $A(x)$.

2.1 Syntax

A two-sided scheme consists of six polynomial time algorithms (Key, DSign, DVerify, USign, Convert, UVerify), and two protocols, a confirmation protocol Confirm and a disavowal protocol Disavow.

Key is a probabilistic algorithm which outputs a public-key pk and a secret-key sk on input 1^ℓ , where ℓ is a security parameter. The public-key pk specifies the message space \mathcal{M} , the space of digital signatures \mathcal{D} , and the space of undeniable signatures \mathcal{U} .

DSign is a (either probabilistic or deterministic) algorithm which outputs a digital signature σ on input (sk, m) , where m is a message. We say that (m, σ) is a valid D-pair if there exists a random tape such that the algorithm $\text{DSign}(sk, m)$ outputs σ .

DVerify is an algorithm which, on input (pk, m, σ) , outputs *accept* if (m, σ) is a valid D-pair, and *reject* otherwise.

USign is a (probabilistic) algorithm which outputs an undeniable signature τ on input (sk, m) , where m is a message. We say that (m, τ) is a valid U-pair if there exists a random tape such that the algorithm $\text{USign}(sk, m)$ outputs τ .

Convert is an algorithm which outputs a digital signature σ for a valid U-pair (m, τ) . More precisely, on input (sk, m, τ) , it outputs *some* $\sigma \leftarrow \text{DSign}(sk, m)$ if (m, τ) is a valid U-pair, and \perp otherwise. Then by using **UVerify** shown below, the validity of (m, τ) is made publicly verifiable.

Note that the above σ is not necessarily a random output of $\text{DSign}(sk, m)$.

It must be related to τ so that the validity of (m, τ) is made publicly verifiable with **UVerify**.

UVerify is an algorithm which verifies the validity of (m, τ) by using $\sigma \leftarrow \text{Convert}(sk, m, \tau)$. More precisely, on input (pk, m, τ, σ) , it outputs *accept* if (m, τ) is a valid U-pair and $\sigma \leftarrow \text{Convert}(sk, m, \tau)$, and *reject* otherwise.

Confirm is a zero-knowledge proof system for valid U-pairs (m, τ) .

Disavow is a zero-knowledge proof system for invalid U-pairs (m, τ) .

A two-sided scheme has three modes as follows.

DS mode: (Key, DSign, DVerify) is used as a DS scheme in an obvious way.

US mode: (Key, USign, Confirm, Disavow) is used as an US scheme in an obvious way.

Selective conversion mode: Convert and UVerify are used to convert an undeniable signature τ on m so that the validity of (m, τ) is made publicly verifiable.

The definitions of Convert and UVerify combine DS mode and US mode through selective conversion mode.

2.2 Security

In two-sided schemes, adversaries have three goals, DS-forgery, US-forgery and invisibility. In the attack game of each goal, we allow \mathcal{A} to have oracle access to DSign-oracle, USign-oracle, Convert-oracle and Confirm/Disavow-oracle, where the last oracle is explained as follows. \mathcal{A} queries (m, τ) to Confirm/Disavow-oracle. If (m, τ) is a valid U-pair, then the oracle returns *yes* and execute the protocol Confirm with \mathcal{A} . Otherwise, it returns *no* and execute the protocol Disavow with \mathcal{A} . In both cases, the oracle plays a role of the signer and \mathcal{A} plays a role of the verifier.

We call DSign-oracle and USign-oracle *sign-oracles*, and Convert-oracle and Confirm/Disavow-oracle *decision-oracles*.

Table 1. Sign-oracles and Decision-oracles

Sign-oracles	DSign-oracle, USign-oracle
Decision-oracles	Convert-oracle, Confirm/Disavow-oracle

(1) We define DS-forgery as follows. Any adversary \mathcal{A} can obtain a valid D-pair (m, σ) if \mathcal{A} queries m to DSign-oracle or \mathcal{A} queries a valid U-pair (m, τ) to Convert-oracle. (In the latter case, Convert-oracle returns σ .) We require that there is no other method for \mathcal{A} to output a valid D-pair. Formally, we consider the following game. An adversary \mathcal{A} is given a randomly generated public-key pk . \mathcal{A} then has access to all oracles. Finally \mathcal{A} outputs a forgery (m^*, σ^*) .

We say that (m^*, σ^*) is not fresh if \mathcal{A} queries m^* to DSign-oracle or \mathcal{A} queries a valid U-pair (m^*, τ) to Convert-oracle for some τ . Otherwise we say that (m^*, σ^*) is fresh. We say that \mathcal{A} DS-forges if (m^*, σ^*) is a valid D-pair, and it is fresh.

We show an example by using Table 2. In this example,

1. \mathcal{A} queried m_i to DSign-oracle and received σ_i .
2. \mathcal{A} queried m_j to USign-oracle and received τ_j . \mathcal{A} next queried (m_j, τ_j) to Convert-oracle and received σ_j .
3. \mathcal{A} queried m_k to USign-oracle and received τ_k .
4. \mathcal{A} queried m_ℓ to no sign-oracle.

\mathcal{A} finally outputs (m^*, σ^*) . If (m^*, σ^*) is a valid D-pair and $m^* = m_\ell$, then \mathcal{A} succeeds in DS-forgery. \mathcal{A} also succeeds even if $m^* = m_k$. It is easy to see that (m^*, σ^*) is fresh in these cases.

Definition 1. We say that a two-sided scheme is DS-secure if $\Pr[\mathcal{A} \text{ DS-forges}]$ is negligible for any PPT adversary \mathcal{A} .

In selective convertible US schemes, \mathcal{A} should not be able to forge a converter α for an already issued U-pair (m, τ) . In two-sided schemes, this security notion is included in the above definition.

(2) We define US-forgery as follows. Suppose that an adversary \mathcal{A} finally outputs a valid U-pair (m^*, τ^*) , where \mathcal{A} has never queried m^* to USign-oracle, but it

queried m^* to DSign-oracle. Is it a forgery? Our definitions of Sec.2.1, however, does not exclude the possibility that one can construct τ^* from a valid D-pair (m^*, σ^*) . Indeed, this is the case in our constructions.

Hence we consider that \mathcal{A} succeeds in US-forgery if \mathcal{A} has never queried m^* to any sign-oracle. We say that a valid U-pair (m^*, τ^*) is fresh if \mathcal{A} has never queried m^* to any sign-oracle. We also consider that \mathcal{A} succeeds in US-forgery if she queries a fresh (m^*, τ^*) to one of the decision oracles during the attack game.

Formally, we consider the following game. An adversary \mathcal{A} is given a randomly generated public-key pk . \mathcal{A} then has access to all oracles. We say that \mathcal{A} US-forges if \mathcal{A} outputs a fresh (m^*, τ^*) or \mathcal{A} queries a fresh (m^*, τ^*) to one of the decision-oracles.

Let's consider the example which is shown in the previous case (1) by using Table 2. Suppose that \mathcal{A} finally outputs a valid U-pair (m^*, τ^*) . If $m^* = m_\ell$, then \mathcal{A} succeeds in US-forgery. However, \mathcal{A} does not succeed if $m^* = m_i$.

Definition 2. We say that a two-sided scheme is US-secure if $\Pr[\mathcal{A} \text{ US-forges}]$ is negligible for any PPT adversary \mathcal{A} .

Table 2. Query pattern and DS/US forgery

	m_i	m_j	m_k	m_ℓ
DSign-oracle	σ_i		(σ^*)	(σ^*)
USign-oracle		τ_j	τ_k	(τ^*)
Convert-oracle		σ_j		

(3) The third security notion is invisibility, a notion due to Chaum, van Heijst and Pfitzmann [9]. This notion is essentially the inability to determine whether a given U-pair is valid. We consider the following game on a distinguisher D .

1. D is given a randomly generated public-key pk . D then has access to all oracles.
2. At some point, D outputs a message m^* which has never been queried to any oracle, and requests a challenge undeniable signature τ^\dagger on m^* .
3. τ^\dagger is generated based on the outcome of a hidden coin toss b . If $b = 1$, then τ^\dagger is generated as usual using USign-oracle, otherwise τ^\dagger is chosen uniformly at random from the undeniable signature space \mathcal{U} .
4. D performs oracle queries again with the restriction that no sign-oracle query on m^* is allowed, and no decision-oracle query on (m^*, τ^\dagger) is allowed.
5. At the end of this attack game, D outputs a guess b' .

$$\text{Define } Adv_{inv}(D) = |\Pr(b' = b) - (1/2)|.$$

Definition 3. A two-sided scheme is invisible if $Adv_{inv}(D)$ is negligible for any PPT D .

Definition 4. We say that a two-sided scheme is secure if it is DS-secure, US-secure and invisible.

3 Proposed Two-Sided Scheme in RO Model

Now we show an efficient two-sided scheme in the random oracle model based on RSA and Paillier’s encryption scheme [29].

3.1 Paillier’s Encryption Scheme

In Paillier’s encryption scheme [29], the public-key is $N(= pq)$, and the private-key is (p, q) , where p and q are large primes. The encryption function for a message $m \in Z_N$ is given by

$$E(m, r) = r^N(1 + mN) \bmod N^2,$$

where $r \in Z_N^*$ is randomly chosen. E has a homomorphic property such that

$$E(m_1, r_1) \cdot E(m_2, r_2) = E(m_1 + m_2 \bmod N, r_1 r_2 \bmod N) \bmod N^2.$$

(For decryption, see [29].) We say that $Y \in Z_{N^2}^*$ is an N th residue if $Y = x^N \bmod N^2$ for some $x \in Z_N^*$. Note that $E(0, r)$ is an N th residue.

3.2 Proposed Scheme

The proposed two-sided scheme is described as follows. Let $m \in \{0, 1\}^*$ be a message.

- **Key Generation.** On input 1^ℓ , choose two primes p, q such that $|p| = |q| = \ell$ randomly and compute $N = pq$. Find d such that $Nd = 1 \bmod lcm(p - 1, q - 1)$. Let $H : \{0, 1\}^* \rightarrow Z_N^*$ be a hash function. Set the public key as $pk = (N, H)$ and the secret key as d .
- **DSign.** Compute $\sigma = H(m)^d \bmod N$ and return σ as the digital signature.
- **DVerify.** For a given (m, σ) , output *accept* if $\sigma^N = H(m) \bmod N$ and *reject* otherwise.
- **USign.** First compute $\sigma = H(m)^d \bmod N$. Next compute τ such that

$$\sigma^N = H(m) + \tau N \bmod N^2. \tag{2}$$

Finally return τ as the undeniable signature.

- **Convert.** For a given (m, τ) , first compute $\sigma = H(m)^d \bmod N$. Next output σ if eq.(2) is satisfied, and \perp otherwise.
- **UVerify.** For a given (m, τ, σ) , output *accept* if eq.(2) is satisfied, and *reject* otherwise.

For confirmation/disavowal protocols, we use the following Lemma.

Lemma 1. (m, τ) is a valid U -pair if and only if there exists $\sigma \in Z_N^*$ such that

$$E(0, \sigma) = H(m) + \tau N \bmod N^2,$$

where E is an encryption function of Paillier’s encryption scheme.

The proof is clear from eq.(2). Now given (m, τ) , the signer computes $\beta \in Z_N$ such that

$$E(\beta, \sigma) = H(m) + \tau N \pmod{N^2}. \tag{3}$$

If $\beta = 0$, then the signer runs a confirmation protocol which proves that $\beta = 0$. Otherwise, the signer runs a disavowal protocol which proves that $\beta \neq 0$.

We will show efficient protocols based on the homomorphic property of Paillier’s encryption scheme [29].

3.3 Confirmation Protocol

We first show a basic confirmation protocol which proves that $\beta = 0$ in eq.(3).

1. The verifier chooses $u, v \in Z_N$ and $w \in Z_N^*$ randomly, and compute

$$y = (H(m) + \tau N)^u E(v, w) \pmod{N^2}.$$

He then sends y to the signer. Note that it holds that for some $r \in Z_N^*$,

$$y = E(0, \sigma)^u E(v, w) = E(0 \times u + v, r) = E(v, r) \pmod{N^2}.$$

2. By using the decryption algorithm of Paillier’s encryption scheme, the signer decrypts y and obtains v' such that $y = E(v', r')$ for some r' . Then she sends v' to the verifier.
3. The verifier accepts if $v' = v$, and rejects otherwise.

Theorem 1. Completeness. *If (m, τ) is a valid U -pair, then the verifier always accepts.*

Soundness. *If (m, τ) is not a valid U -pair, then the verifier rejects with overwhelming probability.*

The proof is given in Appendix A. Finally, we construct a zero-knowledge confirmation protocol as follows, where $commit(x)$ is a commitment function.

1. The verifier sends

$$y = (H(m) + \tau N)^u E(v, w) \pmod{N^2} \tag{4}$$

to the signer, where $u, v \in Z_N$ and $w \in Z_N^*$ are randomly chosen.

2. The signer computes v' such that $y = E(v', r')$, and sends $c = commit(v')$ to the verifier.
3. The verifier reveals u, v, w .
4. The signer checks if eq.(4) holds by using u, v, w . If it holds, then the signer opens $c = commit(v')$. Otherwise, she aborts.
5. The verifier accepts if $v' = v$, and rejects otherwise.

Theorem 2. *The above protocol is zero-knowledge confirmation protocol if (i) $commit(x)$ reveals no information on x , and (ii) the signer cannot find x' such that $commit(x) = commit(x')$.*

The proof will be given in the final version. In the random oracle model, we can use a simple $commit(x)$ shown by Pass [30, Sec.4.1] as follows.

Commit phase. For $x \in Z_N$, Alice chooses $r \in Z_N^*$ randomly and sends $c = H(x, r)$ to Bob.

Reveal phase. Alice sends (x, r) to Bob. Bob checks that $c = H(x, r)$.

3.4 Disavowal Protocol

We first show a basic disavowal protocol which proves that $\beta \neq 0$ in eq.(3).

1. The verifier chooses $u \in Z_N$ and $w \in Z_N^*$ randomly, and computes

$$y = (H(m) + \tau N)^u E(0, w) \bmod N^2.$$

He sends y to the signer. Note that for some $r \in Z_N^*$,

$$y = E(\beta, \sigma)^u E(0, w) = E(\beta \times u \bmod N, r) \bmod N^2. \quad (5)$$

2. The signer first computes x such that $y = E(x, r')$, where $x = \beta \cdot u \bmod N$ from eq.(5). She next computes $u' = x/\beta \bmod N$. Then she sends u' to the verifier.
3. The verifier accepts if $u' = u$, and rejects otherwise.

Similarly to Theorem 1, we can prove the following theorem.

Theorem 3. Completeness. *If (m, τ) is not a valid U -pair, then the verifier always accepts.*

Soundness. *If (m, τ) is a valid U -pair, then the verifier rejects with overwhelming probability.*

Finally we construct a zero-knowledge disavowal protocol as follows, where $commit(x)$ is a commitment function given in the previous subsection.

1. The verifier sends

$$y = (H(m) + \tau N)^u E(0, w) \bmod N^2 \quad (6)$$

to the signer, where $u \in Z_N$ and $w \in Z_N^*$ are randomly chosen.

2. The signer first computes β of eq.(3) and x such that $y = E(x, r')$. She next computes $u' = x/\beta \bmod N$. Then she sends $c = commit(u')$ to the verifier.
3. The verifier reveals u, w .
4. The signer checks if eq.(6) holds by using u, w . If it holds, then the signer opens $c = commit(u')$. Otherwise, she aborts.
5. The verifier accepts if $u' = u$, and rejects otherwise.

Theorem 4. *The above protocol is zero-knowledge disavowal protocol if (i) $commit(x)$ reveals no information on x , and (ii) the signer cannot find x' such that $commit(x) = commit(x')$.*

The proof will be given in the final version.

3.5 Security of Our Scheme

RSA assumption with $e = N$ (N -RSA Problem) claims that given an RSA modulus N and a random $y \in Z_N^*$, it is hard to compute $x \in Z_N^*$ such that $y = x^N \pmod N$. We now define the N^2 -RSA problem as follows. Given an RSA modulus N and a random N th residue $Y \in Z_{N^2}^*$, compute $x \in Z_N^*$ such that $Y = x^N \pmod{N^2}$. The N^2 -RSA assumption claims that the N^2 -RSA problem is hard. We then prove that the proposed scheme is DS-secure under the N^2 -RSA assumption.

Theorem 5. *The proposed scheme is DS-secure under the N^2 -RSA assumption in the random oracle model.*

The proof is given in Appendix B. It uses the techniques of Coron [12] which was also used by [28].

Given an RSA modulus N and a random $y \in Z_N^*$, the computational N th Residuosity (CNR) problem is to find $z \in Z_N$ such that $y + zN = x^N \pmod{N^2}$ for some $x \in Z_N^*$. The CNR assumption claims that the CNR problem is hard. Catalano et al. proved that CNR problem is as intractable as the one-wayness of Paillier cryptosystem [6]. We prove that the proposed scheme is US-secure under the CNR assumption.

Theorem 6. *The proposed scheme is US-secure under CNR assumption in the random oracle model.*

The proof will be given in the final paper.

Let $\text{Residue}_N = \{Y \mid Y = x^N \pmod{N^2} \text{ for some } x \in Z_N^*\}$. Decisional N th Residuosity (DNR) assumption claims that Residue_N and $Z_{N^2}^*$ are indistinguishable. More precisely, we consider the following game between a challenger and a distinguisher D . For a given $N (= pq)$:

1. The challenger chooses a random bit b . If $b = 0$, then he chooses Y from Residue_N randomly. If $b = 1$, then he chooses Y from $Z_{N^2}^*$ randomly. He then gives Y to D .
2. D outputs a bit b' .

Define $\text{Adv}_{dnr}(D) = |\Pr(b' = b) - (1/2)|$. The DNR assumption claims that $\text{Adv}_{dnr}(D)$ is negligible for any PPT distinguisher D . This problem was first addressed in Paillier cryptosystem, namely Paillier cryptosystem is IND-CPA under DNR assumption [29].

We prove that the proposed scheme is invisible under DCR assumption.

Theorem 7. *The proposed scheme is invisible under DNR assumption in the random oracle model.*

The proof will be given in the final paper.

It is easy to see that the following reductions hold for the underlying problems.

1. N -RSA Problem \Rightarrow CNR Problem \Rightarrow DNR Problem,
2. N -RSA Problem \Rightarrow N^2 -RSA Problem,
3. CNR Problem + N^2 -RSA Problem \Rightarrow N -RSA Problem.

4 How to Remove Random Oracle

In this section, we show an efficient two-sided scheme in the standard model. Cramer-Shoup showed an adaptively secure DS scheme under strong RSA assumption in the standard model [13]. It can be seen as a special case of Shamir-Tauman construction [32] which transforms a weakly secure DS scheme (secure against weak non-adaptive chosen message attack) to an adaptively secure one by combining with a trapdoor commitment scheme. In particular, in Cramer-Shoup scheme, a trapdoor commitment scheme is based on GQ identification scheme [15].

Our two-sided scheme is constructed by modifying Cramer-Shoup DS scheme as follows. First, our DSign algorithm is almost the same as Cramer-Shoup DS scheme except that we use two moduli, $N_1(= p_1q_1)$ for GQ-based trapdoor commitment scheme and $N_2(= p_2q_2)$ for a weakly secure signature part, while Cramer-Shoup scheme uses a single modulus. Next our USign algorithm is obtained by extending our technique of Sec.3 to the GQ-based trapdoor commitment scheme.

4.1 Scheme

(Key Generation) Let ℓ be a security parameter.

1. Choose four ℓ -bit primes p_1, q_1, p_2, q_2 randomly such that $p_2 = 2p' + 1$ and $q_2 = 2q' + 1$, where p' and q' are primes. Let $N_1 = p_1q_1$ and $N_2 = p_2q_2$.
2. Choose $h_1 \in Z_{N_1}^*$ and $h_2, x \in QR_{N_2}$ randomly, where QR_N denotes the set of quadratic residues of mod N .
3. Find d such that $N_1d = 1 \pmod{lcm(p_1 - 1, q_1 - 1)}$. Let H be a collision-resistant hash function whose output can be interpreted as a positive integer less than 2^ℓ .
4. Set the public-key as $pk = (N_1, h_1, N_2, h_2, x, H)$ and the secret-key as $sk = (d, p_2, q_2)$.

DSign. For a message $m \in \{0, 1\}^*$, first choose $y' \in Z_{N_1}^*$ randomly and compute $x' \in Z_{N_1}$ such that

$$(y')^{N_1} = x' h_1^{H(m)} \pmod{N_1}, \tag{7}$$

(where x' can be seen as a commitment of m). Next choose a $(\ell + 1)$ -bit prime e randomly and compute y such that

$$y^e = x h_2^{H(x')} \pmod{N_2}, \tag{8}$$

(where (e, y) is a weakly secure signature on x'). The digital signature on m is $\sigma = (e, y, y')$.

DVerify. For a given (m, σ) , first check if e is an $(\ell + 1)$ -bit number. Second, $x' = (y')^{N_1} h_1^{-H(m)} \pmod{N_1}$ is computed. Third, it is checked that $x = y^e h_2^{-H(x')} \pmod{N_2}$.

USign. For a message $m \in \{0, 1\}^*$, first compute $\sigma = (e, y, y')$ as shown in DSign. Next compute $\omega \in Z_{N_1}$ such that

$$(y')^{N_1} = u + \omega N_1 \pmod{N_1^2}, \tag{9}$$

where $u = x'h_1^{H(m)} \pmod{N_1}$. Finally return $\tau = (e, y, x', \omega)$ as the undeniable signature on m . (Note that the above equation is basically the same as eq.(2)).

Convert. For a given m and $\tau = (e, y, x', \omega)$, first check if e is an $(\ell + 1)$ -bit number and (e, y, x') satisfies eq.(8). Next compute $y' \in Z_{N_1}$ which satisfies eq.(7). Finally check if (y', ω) satisfies eq.(7). If everything is OK, then output $\sigma = (e, y, y')$. Otherwise, output \perp .

UVerify. For a given m , $\tau = (e, y, x', \omega)$ and $\sigma = (e, y, y')$, output *accept* if e is an $(\ell + 1)$ -bit number, and eq.(7), eq.(8) and eq.(9) are satisfied, and *reject* otherwise.

In the confirmation protocol, the signer proves that for a valid U-pair, m and $\tau = (e, y, x', \omega)$, there exists $\sigma = (e, y, y')$ which satisfies eq.(7), eq.(8) and eq.(9). Essentially, this means that the signer proves that there exists $y' \in Z_{N_1}$ which satisfies eq.(9). Such a zero-knowledge protocol can be constructed similarly to Sec.3.3.

In the disavowal protocol, the signer proves that for an invalid U-pair m and $\tau = (e, y, x', \omega)$, there exists no $\sigma = (e, y, y')$ which satisfies eq.(7), eq.(8) and eq.(9). If eq.(8) is not satisfied, then we have done. If eq.(8) is satisfied, then the signer proves that there exists no $y' \in Z_{N_1}$ which satisfies eq.(9). Such a zero-knowledge protocol can be constructed similarly to Sec.3.4.

In these protocols, we can use a commitment function based on RSA assumption as shown in [20, Sec.3]. Also, see [18, page 405].

4.2 Security

The strong RSA assumption claims that given an RSA modulus N and a random $y \in Z_N^*$, it is hard to find $e > 1$ and $x \in Z_N^*$ such that $y = x^e \pmod{N}$.

We define the strong CNR problem as follows. Given an RSA modulus N and a random $z \in Z_N^*$, find $a > 1$ and $c \in Z_N$ such that $w = z^a + cN \pmod{N^2}$ is an N th residue. Solving the CNR problem implies an algorithm for solving the strong CNR problem, but the other direction is unknown. The strong CNR assumption claims that the strong CNR problem is hard.

Theorem 8. *The above scheme is US-secure under the strong RSA assumption and the strong CNR assumption in the standard model.*

Theorem 9. *The above scheme is DS-secure under the strong RSA assumption and the strong CNR assumption in the standard model.*

Theorem 10. *The above scheme is invisible under DNR assumption in the standard model.*

All the proofs will be given in the final paper.

References

1. J. Boyar, D. Chaum, I. Damgård and T. Pedersen. Convertible undeniable signatures. *CRYPTO '90*, LNCS 537, pp.189–208, Springer-Verlag, 1990.
2. C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. *ASIACRYPT '98*, LNCS 1514, pp.271–285, Springer-Verlag, 1998.
3. I. Biehl, S. Paulus and T. Takagi. Efficient undeniable signature schemes based on ideal arithmetic in quadratic orders. *Designs, Codes and Cryptography*, Vol. 31, Issue 2, pp.99–123, 2004
4. J. Camenisch and M. Michels. Confirmer signature schemes secure against adaptive adversaries. *EUROCRYPT '00*, LNCS 1870, pp.243–258, Springer-Verlag, 2000.
5. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. *CRYPTO '03*, LNCS 2729, pp.126–144, Springer-Verlag, 2003.
6. D. Catalano, P. Nguyen, J. Stern. The hardness of Hensel lifting: The Case of RSA and Discrete Logarithm. *ASIACRYPT '02*, LNCS 2501, pp.299–310, Springer-Verlag, 2002.
7. D. Chaum. Zero-knowledge undeniable signatures. *EUROCRYPT '90*, LNCS 473, pp.458–464, Springer-Verlag, 1990.
8. D. Chaum. Designated confirmer signatures. *EUROCRYPT '94*, LNCS 950, pp.86–91, Springer-Verlag, 1995.
9. D. Chaum, E. van Heijst and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. *CRYPTO '91*, LNCS 576, pp.470–484, Springer-Verlag, 1991.
10. D. Chaum and H. van Antwerpen. Undeniable signatures. *CRYPTO '89*, LNCS 435, pp.212–216, Springer-Verlag, 1989.
11. T. Chaum and T. P. Pedersen. Wallet databases with observers. *CRYPTO '92*, LNCS 740, pp.89–105, Springer-Verlag, 1993.
12. J. -S. Coron. On the exact security of full domain hash. *CRYPTO '00*, LNCS 1880, pp.229–235, Springer-Verlag, 2000.
13. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, vol.3, no.3, pp.161–185, 2000.
14. I. Damgård and T. Pedersen. New convertible undeniable signature schemes. *EUROCRYPT '96*, LNCS 1070, pp.372–386, Springer-Verlag, 1996.
15. L. Guillou and J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. *EUROCRYPT '88*, LNCS 330, pp.123–128, Springer-Verlag, 1989.
16. S. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. *Topics in Cryptology — CT-RSA '03*, LNCS 2612, pp.80–97, Springer Verlag, 2003.
17. S. Galbraith, W. Mao and K. G. Paterson. RSA-based undeniable signatures for general moduli. *CT-RSA '02*, LNCS 2271, pp. 200–217, Springer Verlag, 2002.
18. R. Gennaro, T. Rabin and H. Krawczyk. RSA-based undeniable signatures. *Journal of Cryptology*, 13(4), pp.397–416, 2000.
19. M. Jakobsson, K. Sako and R. Impagliazzo. Designated verifier proofs and their applications. *EUROCRYPT '96*, LNCS 1070, pp.143–154, Springer-Verlag, 1996.
20. K. Kurosawa and S. Heng: The Power of identification schemes. *PKC '06*, LNCS 3958, pp.364–377, Springer-Verlag, 2006.
21. K. Kurosawa and S. Heng: Relations among security notions for undeniable signature schemes. accepted by SCN 2006.

22. B. Libert and J.-J. Quisquater. Identity based undeniable signatures. *CT-RSA '04*, LNCS 2964, pp.112–125, Springer-Verlag, 2004.
23. F. Laguillaumie and D. Vergnaud: Short undeniable signatures without random oracles: The Missing Link. *INDOCRYPT '05*, LNCS 3797, pp.283–296, Springer-Verlag, 2005.
24. M. Michels, H. Petersen and P. Hoster. Breaking and repairing a convertible undeniable signature scheme. In *3rd ACM CCCS*, pp.148–152, 1996.
25. M. Michels and M. Stadler. Efficient convertible undeniable signature schemes. *SAC '97*, pp.231–244, Springer-Verlag, 1997.
26. J. Monnerat and S. Vaudenay. Undeniable signatures based on characters: how to sign with one bit. *PKC '04*, LNCS 2947, pp.361–396, Springer-Verlag, 2004.
27. J. Monnerat and S. Vaudenay. Generic homomorphic undeniable signatures. *ASIACRYPT '04*, LNCS 3329, pp.354–371, Springer-Verlag, 2004.
28. W. Ogata, K. Kurosawa and S. Heng. The security of the FDH variant of Chaum's undeniable signature scheme. *IEEE Transactions on Information Theory*, 52(5), pp.2006–2017, 2006.
29. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *EUROCRYPT '99*, LNCS 1592, pp.223–238, Springer-Verlag, 1999.
30. R. Pass. On deniability in the common reference string and random oracle model. *CRYPTO '03*, LNCS 2729, pp.316–337, Springer-Verlag, 2003.
31. D. Pointcheval. Self-scrambling anonymizers. *FC '00*, LNCS 1962, pp.259–275, Springer-Verlag, 2000.
32. A. Shamir and Y. Tauman. Improved online/offline signature schemes. *CRYPTO '01*, LNCS 2139, pp.355–367, Springer-Verlag, 2001.

A Proof of Theorem 1

The completeness is clear. We prove the soundness. Suppose that (m, τ) is not a valid U-pair. Then we can write

$$E(\beta, \sigma) = H(m) + \tau N \bmod N^2$$

for some $\beta \in Z_N$ and $\sigma \in Z_N^*$, where $\beta \neq 0$ from Lemma 1. Then y is written as

$$y = E(\beta, \sigma)^u E(v, w) = E(t, r),$$

where

$$t = \beta \cdot u + v \bmod N \text{ and } r = \sigma^u \cdot w \bmod N.$$

Now it is easy to see that for any $v' \in Z_N$, there exists unique $u', w' \in Z_N$ such that

$$t = \beta \cdot u' + v' \bmod N \text{ and } r = \sigma^{u'} \cdot w' \bmod N$$

if $\gcd(\beta, N) = 1$. This means that the prover cannot compute v correctly more than guessing. Hence the verifier rejects with overwhelming probability.

B Proof of Theorem 5

We show that if there exists a PPT adversary \mathcal{A} with $\Pr[\mathcal{A} \text{ DS-forges}] = \epsilon_A$, then one can construct a PPT algorithm M that can solve the N^2 -RSA problem with probability ϵ_M , by running \mathcal{A} as a subroutine. Suppose the input to M is (N, Y) , where $Y = x^N \bmod N^2$ for some $x \in \mathbb{Z}_N^*$.

M then starts running \mathcal{A} by feeding \mathcal{A} with the public key (N, H) where H is a random oracle that will be simulated by M . M also simulates the sign-oracles and the decision-oracles itself.

We assume that when \mathcal{A} requests a sign-oracle query or a decision-oracle query on a message m_i , it has already made the corresponding H query on m_i . When \mathcal{A} makes a H -oracle query for a message m_i , M chooses $r_i \in \mathbb{Z}_N^*$ randomly and behaves as follows.

- With probability δ , return $h_i = H(m_i) = r_i^N \bmod N$. Let $flag_i = 0$, $\sigma_i = r_i$, and compute $\tau_i \in \mathbb{Z}_N^*$ such that $r_i^N = h_i + \tau_i N \bmod N^2$.
- With probability $1 - \delta$, return $h_i = H(m_i) = Yr_i^N \bmod N$. Let $flag_i = 1$, and compute $\tau_i \in \mathbb{Z}_N^*$ such that $r_i^N Y = h_i + \tau_i N \bmod N^2$.

In the above, δ is a fixed probability which will be determined later.

Suppose that \mathcal{A} makes a sign-oracle query for a message m_i .

- Suppose that $flag_i = 0$. If the query is a DSign-oracle query, then M returns σ_i . If it is a USign-oracle query, then M returns τ_i .
- Suppose that $flag_i = 1$. If the query is a USign-oracle query, then M returns τ_i . If the query is a DSign-oracle query, then M aborts and it fails to solve N^2 -RSA problem.

$flag_i$		DSign-oracle query	USign-oracle query
0	$r_i^N = h_i + \tau_i N \bmod N^2$	$\sigma_i = r_i$	τ_i
1	$Yr_i^N = h_i + \tau_i N \bmod N^2$	Abort	τ_i

Next, suppose \mathcal{A} makes a decision-oracle query for (m_i, τ'_i) .

- Suppose that $\tau'_i \neq \tau_i$. If the query is a Convert-oracle query, then M returns \perp . If the query is a Confirm/Disavow-oracle query, then M returns *no* and runs the disavowal protocol with \mathcal{A} .
- Otherwise, $\tau'_i = \tau_i$. If the query is a Confirm/Disavow-oracle query, then M returns *yes* and runs the confirmation protocol with \mathcal{A} .

Suppose that the query is a Convert-oracle query. If $flag_i = 0$, then M returns σ_i . If $flag_i = 1$, then M aborts and it fails to solve N^2 -RSA problem.

In the above, M can simulate the Confirm/Disavow oracle by using the rewinding technique because the protocols are zero-knowledge.

Now suppose that \mathcal{A} DS-forges, and outputs a valid D-pair (m^*, σ^*) at the end of the game. We assume that \mathcal{A} has queried the H -oracle on m^* and so $m^* = m_j$ for some j .

- If $flag_j = 0$, then M aborts.
- Otherwise, $flag_j = 1$. Since (m^*, σ^*) is a valid D-pair, it holds that

$$h_j + \tau_j N = (\sigma^*)^N \bmod N^2.$$

On the other hand, $r_j^N Y = h_j + \tau_j N \bmod N^2$ since $flag_j = 1$. Therefore, it holds that

$$r_j^N Y = (\sigma^*)^N \bmod N^2.$$

$$Y = (\sigma^*/r_j)^N \bmod N^2.$$

Now let $x = \sigma^*/r_j \bmod N$. Then it is easy to show that $x^N = (\sigma^*/r_j)^N \bmod N^2$. Therefore, it holds that

$$Y = x^N \bmod N^2.$$

Consequently, M outputs $x \in Z_N^*$ and thus it solves N^2 -RSA problem.

To complete the proof, it remains to calculate the probability that M does not abort. Let q_D be the number of DSign-oracle queries and that \mathcal{A} issues. The probability that M answers to all DSign-oracle queries is δ^{q_D} , and $flag_j = 1$ for $m_j = m^*$ is $1 - \delta$. Therefore, the probability that M does not abort during the simulation is $\delta^{q_D}(1 - \delta)$. This value is maximized at $\delta_{opt} = 1 - 1/(q_D + 1)$. This shows that ϵ_M is at least $(1/e(1 + q_D))\epsilon_A$, where e is the base of the natural logarithm. This is because the value $(1 - 1/(q_D + 1))^{q_D}$ approaches $1/e$ for large q_S . This completes our proof.

Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures

Jens Groth*

UCLA, Computer Science Department
3531A Boelter Hall
Los Angeles, CA 90095, USA
jg@cs.ucla.edu

Abstract. Non-interactive zero-knowledge proofs play an essential role in many cryptographic protocols. We suggest several NIZK proof systems based on prime order groups with a bilinear map. We obtain linear size proofs for relations among group elements without going through an expensive reduction to an NP-complete language such as Circuit Satisfiability. Security of all our constructions is based on the decisional linear assumption.

The NIZK proof system is quite general and has many applications such as digital signatures, verifiable encryption and group signatures. We focus on the latter and get the first group signature scheme satisfying the strong security definition of Bellare, Shi and Zhang [7] in the standard model without random oracles where each group signature consists only of a constant number of group elements.

We also suggest a simulation-sound NIZK proof of knowledge, which is much more efficient than previous constructions in the literature.

Caveat: The constants are large, and therefore our schemes are not practical. Nonetheless, we find it very interesting for the first time to have NIZK proofs and group signatures that except for a constant factor are optimal without using the random oracle model to argue security.

Keywords: Non-interactive zero-knowledge, simulation-sound extractability, group signatures, decisional linear assumption.

1 Introduction

A non-interactive proof system allows a prover to convince a verifier about the truth of a statement. Zero-knowledge captures the notion that the verifier learns no more from the proof than the truth of the statement. We refer to the full paper [28] for formal definitions of non-interactive zero-knowledge (NIZK) proofs. Our goal in this paper is to construct short efficient prover NIZK proofs for languages that come up in practice when constructing cryptographic protocols. As an example of the usefulness of these new techniques, we construct group signatures consisting of a constant number of group elements.

1.1 Setup

We use two cyclic groups \mathbb{G}, \mathbb{G}_1 of order p , where p is a prime. We make use of a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$. I.e., for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$ we have $e(u^a, v^b) =$

* Supported by NSF grant No. 0456717, and NSF Cybertrust grant.

$e(u, v)^{ab}$. We require that $e(g, g)$ is a generator of \mathbb{G}_1 if g is a generator of \mathbb{G} . We also require that group operations, group membership, and the bilinear map be efficiently computable. Such groups have been widely used in cryptography in recent years.

Let \mathcal{G} be an algorithm that takes a security parameter as input and outputs $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ such that p is prime, \mathbb{G}, \mathbb{G}_1 are descriptions of groups of order p , $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is an admissible bilinear map as described above and g is a random generator of \mathbb{G} .

We use the decisional linear assumption from Boneh, Boyen and Shacham [10].

Definition 1 (Decisional Linear Assumption (DLIN)). *We say the decisional linear assumption holds for the bilinear group generator \mathcal{G} if for all non-uniform polynomial time adversaries \mathcal{A} we have*

$$\begin{aligned} & \Pr \left[(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k); x, y, r, s \leftarrow \mathbb{Z}_p : \right. \\ & \qquad \qquad \qquad \left. \mathcal{A}(p, \mathbb{G}, \mathbb{G}_1, e, g, g^x, g^y, g^{xr}, g^{ys}, g^{r+s}) = 1 \right] \\ \approx & \Pr \left[(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k); x, y, r, s, d \leftarrow \mathbb{Z}_p : \right. \\ & \qquad \qquad \qquad \left. \mathcal{A}(p, \mathbb{G}, \mathbb{G}_1, e, g, g^x, g^y, g^{xr}, g^{ys}, g^d) = 1 \right]. \end{aligned}$$

Throughout the paper, we work over a bilinear group $(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k)$ generated such that the DLIN assumption holds for \mathcal{G} . We call this a DLIN group. Honest parties always check group membership of \mathbb{G}, \mathbb{G}_1 when relevant and halt if an element does not belong to a group that it was supposed to according to the protocol.

Given a DLIN group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ we can set up a semantically secure cryptosystem as in [10]. We choose at random $x, y \leftarrow \mathbb{Z}_p^*$. The public key is (f, h) , where $f = g^x, h = g^y$, and the secret key is (x, y) . To encrypt a message $m \in \mathbb{G}$ we choose $r, s \leftarrow \mathbb{Z}_p$ and let the ciphertext be $(u, v, w) = (f^r, h^s, g^{r+s}m)$. To decrypt a ciphertext $(u, v, w) \in \mathbb{G}^3$ we compute $m = D(u, v, w) = u^{-1/x}v^{-1/y}w$.

The cryptosystem (K_{cpa}, E, D) has several nice properties. The DLIN assumption for \mathcal{G} implies semantic security under chosen plaintext attack (CPA). All triples $(u, v, w) \in \mathbb{G}^3$ are valid ciphertexts. Also, the cryptosystem is homomorphic.

$$E(m_1; r_1, s_1)E(m_2, r_2, s_2) = E(m_1m_2; r_1 + r_2, s_1 + s_2).$$

1.2 Pairing Product Equations

Given a group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ we define a pairing product equation of length ℓ over variables a_1, \dots, a_n to be an equation of the following form.

$$\prod_{j=1}^{\ell} e(q_{j,0}, q_{j,1}) = 1, \text{ where } q_{j,b} = b_{j,b} \prod_{i=1}^n a_i^{e_{j,b,i}} \text{ with } b_{j,b} \in \mathbb{G}, e_{j,b,i} \in \mathbb{Z}_p.$$

Given a set S of pairing product equations $\text{eq}_1, \dots, \text{eq}_m$ we can ask the natural question: *Is there a tuple $(a_1, \dots, a_n) \in \mathbb{G}^n$ such that all equations in S are simultaneously satisfied?*

To illustrate the generality of the language of satisfiable pairing product equations we observe a reduction from the NP-complete language Circuit Satisfiability. Let a_1, \dots, a_n correspond to the wires of the circuit, which without loss of generality contains only NAND-gates. Let S contain equations $e(a_i, a_i g^{-1}) = 1$ forcing each $a_i = g^{b_i}$ to encode a bit $b_i \in \{0, 1\}$. For each NAND-gate with input wires i_0, i_1 and output i_2 add to S the equation $e(a_{i_0}, a_{i_1}) = e(g, g a_{i_2}^{-1})$, which is satisfied if and only if $b_{i_2} = \neg(b_{i_0} \wedge b_{i_1})$.

Our main motivation for being interested in satisfiability of pairing product equations is not NP-completeness though. Satisfiability of pairing product equations comes up in practice when constructing cryptographic protocols and by making a direct NIZK proof instead of first reducing the problem to some other language such as Circuit Satisfiability we keep proofs short.

For concreteness, let us use verifiable encryption as an example of a pairing product satisfiability question that may come up in practice. Suppose (u, v, w) is a ciphertext under the public key (f, h) of the DLIN-based cryptosystem described earlier. We are interested in whether this ciphertext encrypts a particular message m . This is the case, if and only if there exists a such that $e(g, u) = e(a, f)$ and $e(h, w m^{-1} a^{-1}) = e(v, g)$. If we know r, s we can compute the satisfiability witness $a = g^r$.

1.3 NIZK Proofs for Satisfiability of Pairing Product Equations

NIZK PROOFS. The central technical contribution of this paper is an NIZK proof of size $\mathcal{O}(n + \ell)$ group elements for satisfiability of a set of pairing product equations of combined length $\ell = \sum_{j=1}^m \ell_j$. The proof system has perfect completeness and perfect soundness.

RELATED WORK ON NIZK PROOFS. NIZK proofs were introduced by Blum, Feldman and Micali [9] and they suggested an NIZK proof for a single statement based on the hardness of deciding quadratic residuosity. Blum et al. [8] extended this to multi-theorem NIZK proofs. Feige, Lapidot and Shamir [25] and Kilian and Petrank [33] give constructions based on trapdoor permutations.

Recently Groth, Ostrovsky and Sahai [30] have constructed NIZK proofs from composite order bilinear groups introduced by Boneh, Goh and Nissim [11]. Even more recently Groth, Ostrovsky and Sahai [29] have introduced the setting in this paper, a bilinear group of prime order and the DLIN assumption. They construct non-interactive witness-indistinguishable proofs without any setup assumptions. In the common reference string (CRS) model both results give NIZK proofs for Circuit Satisfiability of size $\mathcal{O}(|C|)$ group elements.

All the above-mentioned papers have in common that they focus on an NP-complete language, usually Circuit Satisfiability, and suggest a bit-by-bit or gate-by-gate NIZK proof for this language. Our paper differs by introducing new techniques that allows making *direct* NIZK proofs for satisfiability of pairing product equations. This allows us to construct constant/linear size cryptographic protocols for digital signatures, RCCA-secure encryption[20], verifiable encryption and group signatures.

The only other way we know of to get linear size NIZK proofs/arguments for any practical language is the Fiat-Shamir heuristic: Make a 3-move public coin (honest verifier) zero-knowledge protocol non-interactive by computing the verifier's challenge as

a hash of the statement and the initial protocol message. To argue security, one models the hash-function as a random oracle [6]. It is well known that using the random oracle model sometimes results in insecure real life protocols [18,19,34,27,4]. In comparison, our NIZK proofs have *provable security* under the DLIN assumption.

SIMULATION-SOUND EXTRACTABLE NIZK PROOFS. Combining the definitions of simulation-soundness introduced by Sahai [35] and proofs of knowledge from De Santis and Persiano [23], we get simulation-sound extractability. Here the simulator first creates a simulated CRS together with a simulation trapdoor and an extraction trapdoor. We require that even after the adversary has seen simulated proofs on arbitrary statements, if it constructs a new valid proof on any statement, then we can extract a witness. Simulation-sound extractability is a very strong notion, in particular it implies non-malleability as defined by De Santis et al. [22].

We construct a simulation-sound extractable NIZK proof for satisfiability of pairing product equations. Our NIZK proof has a CRS with a description of the group and a constant number of group elements, and the proofs consist of $\mathcal{O}(n + \ell)$ group elements.

RELATED WORK ON SIMULATION-SOUND NIZK PROOFS. As stated before, our interest in this paper is satisfiability of pairing products equations. However, in order to compare our scheme with previous work let us look at the case of Circuit Satisfiability. [35] constructed a one-time simulation-sound NIZK proof system using techniques from Dwork, Dolev and Naor [24]. Later a construction for unbounded simulation-sound extractable NIZK arguments was given by [22], where the adversary can see many simulated arguments of arbitrary statements. The schemes from both these papers are based on trapdoor permutations but are not practical. For the sake of fairness in evaluating the quality of our contribution, we have also considered whether the techniques from [30] could be used to get good efficiency for simulation-sound extractability. The answer to this question seems to be negative, the best construction we can think of using GOS-techniques gives an additive polynomial size overhead.

Scheme	NIZK proof bit size	Assumption
[22]	$\mathcal{O}(C \text{poly}(k))$	Trapdoor permutations
Potential use of [30] techniques	$\mathcal{O}(C k + \text{poly}(k))$	Subgroup decision
This paper	$\mathcal{O}(C k)$	DLIN

Fig. 1. Comparison of simulation-sound extractable proofs for Circuit Satisfiability

COMMON REFERENCE STRING VERSUS UNIFORM RANDOM STRING. We will construct NIZK proofs and simulation-sound extractable NIZK proofs in the common reference string model, where the prover and the verifier both have access to a CRS chosen according to some distribution. If this distribution is uniform at random we call it the uniform random string model. In some settings it is easier to work with a URS, for instance a URS can easily be jointly generated using multi-party computation techniques.

Our NIZK proofs use a common reference string that contains a description of a bilinear group and a number of group elements. Depending on the group elements, the CRS will give either perfect soundness or perfect zero-knowledge. With overwhelming

probability random group elements will lead to a perfect soundness CRS. Assuming that we can use a uniform random string to get a description of a DLIN group and a number of random group elements, we will therefore get NIZK proofs and simulation-sound NIZK proofs in the URS-model. Since there is a negligible chance of picking a perfect zero-knowledge CRS, this gives statistical soundness instead of perfect soundness, which is the best we can hope for in the URS-model. We remark that natural candidates for bilinear DLIN groups based on elliptic curves are efficiently samplable from a URS [29]. For the sake of simplicity we will just work with the CRS-model in the paper, but invite the reader to note that all constructions work in the URS-model as well.

1.4 An Application: Constant Size Group Signatures

Group signatures, introduced by Chaum and van Heyst [21], allow a member to sign messages anonymously on behalf of a group. A group manager controls the group and decides who can join. In case of abuse, the group manager is able to open a signature to reveal who the signer is. It is hard to design group signatures and most schemes [17,16,3,14,2,13,31,15,10,26,32] use the random oracle model in the security proof.

Bellare, Micciancio and Warinschi [5] suggest rigorous security definitions for group signatures in the *static* case where the set of members is fixed from the start and never changes. Bellare, Shi and Zhang [7] extend the security model to the partially *dynamic* case where the group manager can enroll new members in the group. Both [5] and [7] suggest constructions of group signatures based on trapdoor permutations. These constructions are very inefficient and only indicate feasibility.

Boyer and Waters [12] use a combination of the Waters signature scheme [36] and the [30] NIZK proofs. They assume a static setting and as part of a group signature they encrypt the identity of the signer bit by bit. This means that a group signature consists of $\mathcal{O}(\log n)$ group elements, where n is the number of members in the group. The group signature scheme satisfies a relaxed version of the [5] security definition, where the anonymity is guaranteed only when no signatures have been opened and traced to the signer. In comparison, the full-anonymity definition in [5] demands that anonymity is preserved even when the adversary can get an opening of any other signature than the challenge.

Ateniese et al. [1] use a bilinear group of prime order. The advantage of this scheme is that it is very efficient, a group signature consists of 8 group elements. However, they use several strong security assumptions and their security model is even weaker than that of [12] since it does not protect against key-exposures; knowledge of a signing key immediately allows one to tell which signatures this member has made. In comparison, the BMW,BSZ-models do guard against key exposure.

The tools in this paper give a construction of group signatures where both keys and signatures consist of a constant number of group elements. The construction involves carefully constructing and tailoring a signature scheme and the simulation-sound extractable NIZK proof system such that they fit each other. The constant is large; we do not claim this to be a practical scheme. Rather this should be seen as an interesting feasibility result; under a simple and natural security assumption there exists an up to a constant optimal dynamic group signature scheme satisfying the strong security definitions from [5,7].

Scheme	Signature in bits	Security model	Assumption
[5]	$\text{poly}(k)$	BMW [5] (fixed group)	Trapdoor permutations
[7]	$\text{poly}(k)$	BSZ [7] (dynamic group)	Trapdoor permutations
[12]	$3k + 2k \log n$	BMW [5], CPA-anonymity	Subgroup decision and CDH
[1]	$8k$	UC-model, non-adaptive adv.	Strong SXDH, q-EDH, strong LRSW
This paper	$\mathcal{O}(k)$	BSZ [7]	DLIN

Fig. 2. Comparison of group signature schemes

2 Preliminaries

2.1 Definitions: Non-interactive Zero-Knowledge Proofs

We provide formal definitions of non-interactive proofs, perfect completeness, perfect soundness, unbounded adaptive zero-knowledge, composable zero-knowledge, perfect proofs of knowledge, simulation soundness and simulation-sound extractability in the full paper. Here we will just sketch one useful stronger definition of zero-knowledge that we have not seen elsewhere in the literature.

COMPOSABLE ZERO-KNOWLEDGE. We define composable zero-knowledge by making two requirements. First, a real CRS is computationally indistinguishable from a simulated CRS; we call this reference string indistinguishability. Second, the adversary *even when it gets access to the simulation trapdoor* τ , cannot distinguish real proofs on the simulated CRS from simulated proofs. We call this simulation indistinguishability. We refer to the full paper for the formal definition and a proof that composable zero-knowledge implies the standard notion of unbounded adaptive zero-knowledge usually found in the literature.

Our motivation for introducing the notion of composable zero-knowledge is that it allows different zero-knowledge proofs for *different* languages to use the *same* CRS. Suppose we have relations R_1, \dots, R_n and corresponding NIZK proof systems $(K, P_1, V_1), \dots, (K, P_n, V_n)$ with composable zero-knowledge using the same key generator and CRS simulator K, S_1 . A hybrid argument shows that no non-uniform polynomial time adversary can distinguish real proofs on a simulated CRS from simulated proofs on this CRS for relation R_i , *even if it sees arbitrary proofs or simulations for statements in $L_{j \neq i}$ using the same CRS*. The reason is that in the definition of simulation indistinguishability we give τ to the adversary, so it can itself implement the simulator $S_{2,j}$ for any relation $R_{j \neq i}$.

Composable zero-knowledge implies that the zero-knowledge property still makes sense when many different NIZK proofs use the same CRS. In our paper, all the NIZK proofs will indeed generate the CRS in the same way and simulate the CRS in the same way, so we get better performance by not having to deal with different CRSs for each proof system. At the same time, it simplifies the paper.

2.2 A Homomorphic Commitment Scheme

We use the cryptosystem from Section 1.1 to create a homomorphic commitment scheme such that depending on how we generate the public key we get either a perfectly binding commitment scheme or a perfectly hiding trapdoor commitment scheme.

The idea is that if K is an encryption of 1, then $K^m E(1; r, s)$ is also an encryption of 1 and we have a perfectly hiding commitment to m . On the other hand, if K is not an encryption of 1, then $K^m E(1; r, s)$ is perfectly binding.

Perfectly binding key generation: Let $ck = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$ where f, h is a public key for the cryptosystem and $(u, v, w) = (f^{r_u}, h^{s_v}, g^{t_w})$ with $t_w \neq r_u + s_v$ is an encryption of a non-trivial element.

Perfectly hiding trapdoor key generation: Let $ck = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$ where f, h is a public key for the cryptosystem and $(u, v, w) = (f^{r_u}, h^{s_v}, g^{r_u+s_v})$ is an encryption of 1.

The corresponding trapdoor key is $tk = (ck, x, y, r_u, s_v)$.

Commitment: To commit to message $m \in \mathbb{Z}_p$ pick $r, s \leftarrow \mathbb{Z}_p$ and let the commitment be $c = (c_1, c_2, c_3) = \text{com}(m; r, s) = (u^m f^r, v^m h^s, w^m g^{r+s})$.

The commitment schemes $(K_{\text{binding}}, \text{com})$ and $(K_{\text{hiding}}, \text{com})$ have several nice properties. The CPA-security of the cryptosystem implies that one cannot distinguish perfect binding keys from perfect hiding keys. This in turn implies computational hiding respectively computational binding for the two schemes. The homomorphic property of the cryptosystem transfers to the commitment scheme.

$$\text{com}(m_1 + m_2; r_1 + r_2, s_1 + s_2) = \text{com}(m_1; r_1, s_1) \text{com}(m_2; r_2, s_2).$$

For the perfectly binding commitment scheme, any $c \in \mathbb{G}^3$ is a commitment to some message $m \in \mathbb{Z}_p$.

3 Efficient Non-interactive Zero-Knowledge Proof Systems

The construction of our NIZK proof for satisfiability of pairing product equations is very complex and requires many new techniques. We will therefore build it in a modular fashion from NIZK proofs for simpler relations. Even some of these simpler NIZK proofs are complex and we can only sketch the ideas behind the constructions here. The full paper [28] contains full constructions and security proofs.

3.1 Common Reference String

All the NIZK proofs in this section use the same CRS generator K and CRS simulator S_1 described below. A CRS is a public key for the perfectly binding commitment scheme described in the previous section. The soundness of the NIZK proofs comes from the perfect binding property of the commitment scheme, which makes it impossible for any adversary to cheat. In simulations, we use a public key for the perfectly hiding commitment scheme as the simulated CRS.

Common reference string

Generate $\sigma = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w) \leftarrow K_{\text{binding}}(1^k)$.¹

¹ Both the CRS generator K and the CRS simulator S_1 first create a DLIN group honestly. This means that instead of generating the CRSs from scratch, it is also possible to build any of the NIZK proofs we construct in the following sections on top of an already existing DLIN group. When doing so we write $\sigma \leftarrow K(p, \mathbb{G}, \mathbb{G}_1, e, g)$ or $(\sigma, \tau) \leftarrow S_1(p, \mathbb{G}, \mathbb{G}_1, e, g)$.

Simulated reference string

Generate $(\sigma, \tau) \leftarrow K_{\text{hiding}}(1^k)$, where $\sigma = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$ and $\tau = (x, y, r_u, s_u)$.

The CPA-security of the cryptosystem gives us the following lemma.

Lemma 1. *If $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ is a DLIN group, then (K, S_1) has reference string indistinguishability.*

3.2 NIZK Proofs for Commitment to 0

Let $R_{\text{zero}} = \{(c, (r, s)) \mid c = \text{com}(0; r, s)\}$ define the language of commitments to 0. The proof of the following theorem can be found in the full paper.

Theorem 1. *There exists an NIZK proof system $(K, P_{\text{zero}}, V_{\text{zero}}, S_1, S_{\text{zero}})$ for R_{zero} with perfect completeness, perfect soundness and composable zero-knowledge with perfect simulation indistinguishability under the DLIN assumption for \mathcal{G} . The proof consists of 1 group element ($\pi = g^r$). Verification corresponds to evaluating two pairing product equations.*

3.3 Proof for Committed Multiplicative Relationship

Consider three commitments c_a, c_b, c_c such that the corresponding messages have a multiplicative relationship $m_c = m_a m_b$. The corresponding relation is $R_{\text{mult}} = \{(c_a, c_b, c_c), (m_a, r_a, s_a, m_b, r_b, s_b, r_c, s_c) \mid c_a = \text{com}(m_a; r_a, s_a), c_b = \text{com}(m_b; r_b, s_b), c_c = \text{com}(m_a m_b; r_c, s_c)\}$.

Theorem 2. *There exists an NIZK proof $(K, P_{\text{mult}}, V_{\text{mult}}, S_1, S_{\text{mult}})$ for R_{mult} with perfect completeness, perfect soundness and composable zero-knowledge if the DLIN assumption holds for \mathcal{G} . A proof consists of 36 group elements. Verification corresponds to evaluating a set of pairing product equations.*

Sketch of proof. c_a, c_b, c_c have a multiplicative relationship if and only if

$$c_c = c_b^{m_a} \text{com}(0; r_c - m_a r_b, s_c - m_a s_b).$$

To prove the latter, it suffices to reveal m_a , and prove that $c_a \text{com}(-m_a; 0, 0)$ and $c_c c_b^{-m_a}$ are commitments to 0. To get zero-knowledge, we tweak this idea in a way such that m_a is not revealed directly.

The main trick in the NIZK proof is to pick exponents r, s at random, which will be used to hide m_a . Using $(K, P_{\text{zero}}, V_{\text{zero}})$ we prove that

$$\begin{aligned} & c_a \text{com}(1; 0, 0)^{-(r+s+m_a)} (\text{com}(1; 0, 0) \pi_{0,1})^r (\text{com}(1; 0, 0) \pi_{0,3})^s \\ \text{and} \quad & c_c c_b^{-(r+s+m_a)} (c_b \pi_{0,2})^r (c_b \pi_{0,4})^s \end{aligned}$$

are commitments to 0, where $\pi_{0,1}, \pi_{0,2}, \pi_{0,3}, \pi_{0,4}$ are themselves commitments to 0.

Revealing the components $\text{com}(1; 0, 0)^{r+s+m_a}, c_b^{r+s+m_a}$, the verifier can use the bilinear maps to check that there exists some common exponent $t = r + s + m_a$, even

though it cannot compute the exponent itself. Similarly, revealing $(\text{com}(1; 0, 0)\pi_{0,1})^r$, $(c_b\pi_{0,2})^r$ and $(\text{com}(1; 0, 0)\pi_{0,3})^s$, $(c_b\pi_{0,4})^s$ allows the verifier to check that there exist common exponents r, s .

We are verifiably using the same exponents r, s, t on $\text{com}(1; 0, 0)$ and c_b to get respectively c_a and c_c . This shows that

$$c_a \text{com}(1; 0, 0)^{r+s-t} \quad \text{and} \quad c_c c_b^{r+s-t}$$

are both commitments to 0. The only way this can be possible is when $m_a = t - r - s$.

Computational simulation indistinguishability follows from the fact that while we use the same exponents, we use different bases. Therefore, at no point is any element itself raised to m_a , which the adversary could potentially use to detect whether it was a correct proof or one created by a simulator, which does not know m_a . The commitments $\pi_{0,1}, \pi_{0,2}, \pi_{0,3}, \pi_{0,4}$ rerandomize the bases that we raise to r, s and therefore $t = r + s + m_a$ is indistinguishable from t random, so m_a is hidden. \square

3.4 NIZK Proof for Commitment to Exponent

We have two elements a, b and a commitment c to the exponent m so $b = a^m$. $R_{\text{expo}} = \{((a, b, c), (m, r, s)) \mid b = a^m, c = \text{com}(m; r, s)\}$ defines the language of such statements.

Theorem 3. *There exists an NIZK proof $(K, P_{\text{expo}}, V_{\text{expo}}, S_1, S_{\text{expo}})$ for R_{expo} with perfect completeness, perfect soundness and composable zero-knowledge with perfect simulation indistinguishability if the DLIN assumption holds for \mathcal{G} . A proof consists of 8 group elements. Verification consists of evaluating a set of pairing product equations.*

Sketch of proof. If $a \neq 1$ then one can use the bilinear map to verify that a pair of commitments π_1, π_m have the same exponent m so $\pi_m = \pi_1^m$. If π_1 is a commitment to 1, then π_m is a commitment to m . What remains is to prove that $\pi_1 \text{com}(-1; 0, 0)$ and $c_m \pi_m^{-1}$ are commitments to 0, which we can do with the NIZK proof for commitment to 0.

To prove zero-knowledge we observe that on a perfect hiding key ck

$$\pi_1 = (a^{xr_1}, a^{ys_1}, a^{r_1+s_1}) \quad \text{and} \quad \pi_m = (b^{xr_1}, b^{ys_1}, b^{r_1+s_1})$$

gives us commitments so $\pi_m = \pi_1^m$, even though we do not know m itself. \square

3.5 NIZK Proof for Generalized Pedersen Commitment

Consider a Pedersen commitment to many messages $b = g^t \prod_{i=1}^n a_i^{m_i}$. Let c_t, c_1, \dots, c_n be commitments to the exponents. The language of multi-message Pedersen commitments and corresponding exponent-commitments is defined by $R_{\text{m-ped}} = \{((a_1, \dots, a_n, b, c_t, c_1, \dots, c_n), (t, r_t, s_t, m_1, r_1, s_1, \dots, m_n, r_n, s_n)) \mid b = g^t \prod_{i=1}^n a_i^{m_i}, c_t = \text{com}(t; r_t, s_t), c_i = \text{com}(m_i, r_i, s_i)\}$.

Theorem 4. *There exists an NIZK proof $(K, P_{\text{m-ped}}, V_{\text{m-ped}}, S_1, S_{\text{m-ped}})$ for $R_{\text{m-ped}}$ with perfect completeness, perfect soundness and composable zero-knowledge if the DLIN assumption holds for \mathcal{G} . The proof consists of $63n - 4$ group elements. The verification consists of evaluating a set of pairing product equations.*

Sketch of Proof. The hard part in constructing an NIZK proof for $R_{m\text{-ped}}$ is to construct a proof for the one-message Pedersen commitment relation R_{ped} , which is done with techniques related to the NIZK proof for multiplicative relationship, see the full paper for details. Once we have that, we split b into n one-message Pedersen commitments $b = \prod_{i=1}^n b_i = \prod_{i=1}^n (a_i^{m_i} g^{t_i})$ choosing the t_i 's at random so $t = \sum_{i=1}^n t_i$ and make commitments c_{t_i} to the t_i 's. We make an NIZK proof for R_{ped} for each of the statements (a_i, b_i, c_i, c_{t_i}) . \square

3.6 NIZK Proof for Committed Bilinear Product

We can commit to $a_1, b_1, \dots, a_n, b_n$ in the following way. We form $A_i = g^{r_i} a_i$ and commitments c_{r_i} to r_i . Similarly, we form $B_i = g^{s_i} b_i$ and commitments c_{s_i} to s_i . We are interested in knowing whether $\prod_{i=1}^n e(a_i, b_i) = 1$.

Let $R_{\text{bil-prod}} = \{(A_1, c_{r_1}, B_1, c_{s_1}, \dots, A_n, c_{r_n}, B_n, c_{s_n}), (r_1, r_{r_1}, s_{r_1}, s_1, r_{s_1}, s_{s_1}, \dots, r_n, r_{r_n}, s_{r_n}, s_n, r_{s_n}, s_{s_n}) \mid A_i = g^{r_i} a_i, B_i = g^{s_i} b_i, c_{r_i} = \text{com}(r_i; r_{r_i}, s_{r_i}), c_{s_i} = \text{com}(s_i; r_{s_i}, s_{s_i}), \prod_{i=1}^n e(a_i, b_i) = 1\}$.

Theorem 5. *There exists an NIZK proof $(K, P_{\text{bil-prod}}, V_{\text{bil-prod}}, S_1, S_{\text{bil-prod}})$ for $R_{\text{bil-prod}}$ with perfect completeness, perfect soundness and composable zero-knowledge under the DLIN assumption for \mathcal{G} . Proofs consist of $228n - 3$ group elements and verification corresponds to evaluating a set of pairing product equations.*

Sketch of proof. The key observation in the construction is that if and only if $\prod_{i=1}^n e(a_i, b_i) = 1$, we have for arbitrary $R_1, S_1, \dots, R_n, S_n \in \mathbb{Z}_p$ that

$$\begin{aligned} \prod_{i=1}^n e(A_i, B_i) &= \prod_{i=1}^n e(g^{r_i}, g^{s_i} b_i) e(g^{r_i} a_i, g^{s_i}) e(g^{r_i}, g^{s_i})^{-1} \prod_{i=1}^n e(a_i, b_i) \\ &= \prod_{i=1}^n e(g, B_i)^{r_i} e(A_i, g)^{s_i} e(g, g)^{-r_i s_i} = e(g, g^{-\sum_{i=1}^n r_i s_i} \prod_{i=1}^n A_i^{s_i} B_i^{r_i}) \\ &= e(g, g^{-\sum_{i=1}^n (r_i s_i + R_i S_i)} \prod_{i=1}^n A_i^{s_i} B_i^{r_i}) \prod_{i=1}^n e(g^{R_i}, g^{S_i}). \end{aligned}$$

In the NIZK proof, we pick $R_1, S_1, \dots, R_n, S_n$ at random. We commit to R_i, S_i and we already have commitments to r_i, s_i . We reveal the $2n+1$ elements $g^{R_1}, g^{S_1}, \dots, g^{R_n}, g^{S_n}$ and $g^{-\sum_{i=1}^n (r_i s_i + R_i S_i)} \prod_{i=1}^n A_i^{s_i} B_i^{r_i}$. We then use NIZK proofs for $R_{\text{expo}}, R_{\text{mult}}, R_{m\text{-ped}}$ to prove that they have been formed correctly.

In the simulation, we observe that for arbitrary $R_1, S_1, \dots, R_n, S_n$

$$\prod_{i=1}^n e(A_i, B_i) = e(g, g^{-\sum_{i=1}^n R_i S_i} \prod_{i=1}^n A_i^{-S_i} B_i^{-R_i}) \prod_{i=1}^n e(g^{R_i} A_i, g^{S_i} B_i).$$

Picking $R_1, S_1, \dots, R_n, S_n$ randomly means all elements have the same distribution as in a real proof on a simulated CRS. We can then simulate the NIZK proofs for $R_{\text{expo}}, R_{\text{mult}}, R_{m\text{-ped}}$. \square

3.7 NIZK Proof for Satisfiability of Pairing Product Equations

Recall from the introduction that a pairing product equation is of the form

$$\text{eq}(a_1, \dots, a_n) : \prod_{j=1}^{\ell} e(q_{j,0}, q_{j,1}) = 1, \text{ where } q_{j,b} = b_{j,b} \prod_{i=1}^n a_i^{e_{j,b,i}},$$

for known $b_{j,b} \in \mathbb{G}$ and $e_{j,b,i} \in \mathbb{Z}_p$. A set S of pairing product equations $\text{eq}_1, \dots, \text{eq}_m$ is said to be satisfiable if there exists $(a_1, \dots, a_n) \in \mathbb{G}^n$ such that all equations are satisfied. Let $R_{\text{ppsats}} = \{ S \mid \exists (a_1, \dots, a_n) \in \mathbb{G}^n \forall \text{eq}_k \in S : \text{eq}_k(a_1, \dots, a_n) = \text{true} \}$. We conclude this section with the following main theorem.

Theorem 6. *There exists an NIZK proof $(K, P_{\text{ppsats}}, V_{\text{ppsats}}, S_1, S_{\text{ppsats}})$ for R_{ppsats} with perfect completeness, perfect soundness and composable zero-knowledge if the DLIN assumption holds for \mathcal{G} . Proofs consist of $4n + 228\ell - 3m$ group elements, where $\ell = \sum_{k=1}^m \ell_k$. Verification consists of evaluating a set of pairing product equations.*

Sketch of proof. In the NIZK proof, we first commit to each a_i as $g^{t_i} a_i$ and $\text{com}(t_i)$. Using homomorphic properties, it is straightforward for $q_{k,j,b}$ in equation eq_k to compute $g^{t_{k,j,b}} q_{k,j,b}$ and $\text{com}(t_{k,j,b})$ as

$$b_{k,j,b} \prod_{i=1}^n (g^{t_i} a_i)^{e_{k,j,b,i}} = g^{\sum_{i=1}^n t_i e_{k,j,b,i}} (b_{k,j,b} \prod_{i=1}^n a_i^{e_{k,j,b,i}})$$

$$\text{and } \prod_{i=1}^n \text{com}(t_i)^{e_{k,j,b,i}} = \text{com}\left(\sum_{i=1}^n t_i e_{k,j,b,i}\right).$$

For each pairing product equation eq_k make an NIZK proof for $R_{\text{bil-prod}}$ that $\prod_{j=1}^{\ell_k} e(q_{k,j,0}, q_{k,j,1}) = 1$. □

NESTING NIZK PROOFS. Since verification consists of verifying a set of pairing product equations, we can nest NIZK proofs inside one another. I.e., we can prove that there exists an NIZK proof such that there exists an NIZK proof such that, etc. Each level of nesting costs a constant blow-up factor. In comparison, this is very expensive with other NIZK proofs and impossible in the random oracle model.

REDUCING THE NUMBER OF VARIABLES. Consider a set of pairing product equations over n variables with combined length ℓ . We show in the full paper that there is a set of pairing product equations of length ℓ over $n' \leq 2\ell$ variables, such that this set is satisfiable if and only if the original set is satisfiable. This gives us NIZK proofs of length $\mathcal{O}(\ell)$ group elements for satisfiability of pairing product equations.

4 Simulation-Sound Extractable NIZK Proof for Satisfiability of Pairing Product Equations

A CMA-SECURE SIGNATURE SCHEME. With the help of the NIZK proof for R_{ppsats} , we can construct a digital signature scheme secure against adaptive chosen message attack (CMA).

Theorem 7. *Under the DLIN assumption there exists a CMA-secure digital signature scheme $(K_{\text{sign}}, \text{Sign}, \text{Ver})$ for signing n group elements with perfect correctness. The verification key and the signatures consist of $\mathcal{O}(n)$ group elements and the verification process consists of evaluating a set of pairing product equations.*

Due to lack of space we refer the reader to the full paper [28] for the construction and the proof. We remark on one issue that makes the construction non-trivial. Our NIZK proofs work for pairing product equations. Since we want to use the NIZK proofs on encrypted signatures, we cannot use a hash-function in the signature scheme, since we do not know how to make NIZK proofs for correct hashing without an expensive NP-reduction to e.g. Circuit Satisfiability.

SIMULATION-SOUND EXTRACTABLE NIZK PROOFS. We will combine the CMA-secure signature scheme with the NIZK proofs to construct an unbounded simulation-sound extractable NIZK proof for R_{ppsat} .

Common reference string and simulated reference string: Given a group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ pick CMA-secure signature keys $(vk, sk) \leftarrow K_{\text{sign}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$, keys for the CPA-secure cryptosystem $(pk, sk_{\text{cpa}}) \leftarrow K_{\text{cpa}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$ and make a ciphertext $c_1 \leftarrow E_{pk}(t)$ for $t \neq 1$. Let $\sigma \leftarrow K(p, \mathbb{G}, \mathbb{G}_1, e, g)$ be a CRS for our NIZK proofs.

The CRS is $\Sigma = (vk, pk, c_1, \sigma)$.

In the simulation we pick $c_1 = E_{pk}(1; r_c, s_c)$ and let the simulation trapdoor be $\tau = (sk, r_c, s_c)$ while the extraction key is $\xi = sk_{\text{cpa}}$.

Proof: Given a set of pairing product equations S and a satisfiability witness $w = (a_1, \dots, a_n)$ the proof is constructed as follows.

Pick keys $(vk_{\text{sots}}, sk_{\text{sots}})$ for a strong one-time signature scheme.² Encrypt $c_w \leftarrow E_{pk}(a_1, \dots, a_n)$ and $c_s = E_{pk}(1, \dots, 1)$. Make an NIZK proof π_{ssor} of the following statement: Either c_w contains a satisfying witness, or c_1 contains 1 and c_s contains a signature under vk on vk_{sots} . We refer to the full paper how to use the NIZK proof for R_{ppsat} to prove satisfiability of at least one out of two sets of pairing product equations. Finally, sign everything $s_{\text{sots}} \leftarrow \text{Sign}_{sk_{\text{sots}}}(S, c_w, c_s, \pi_{\text{ssor}})$.

The proof is $\pi = (vk_{\text{sots}}, c_w, c_s, \pi_{\text{ssor}}, s_{\text{sots}})$.

Simulation: Pick keys $(vk_{\text{sots}}, sk_{\text{sots}})$ for a strong one-time signature scheme. Sign vk_{sots} as $s \leftarrow \text{Sign}_{sk}(vk_{\text{sots}})$. Encrypt $c_w \leftarrow E_{pk}(1, \dots, 1)$ and $c_s = E_{pk}(s)$. Make an NIZK proof π_{ssor} of the following statement: Either c_w contains a satisfying witness, or c_1 contains 1 and c_s contains a signature under vk on vk_{sots} . Finally, sign everything $s_{\text{sots}} \leftarrow \text{Sign}_{sk_{\text{sots}}}(S, c_w, c_s, \pi_{\text{ssor}})$.

Verification and extraction: Accept the proof if and only if the strong one-time signature s_{sots} and the proof π_{ssor} are valid.

To extract a witness simply decrypt c_w .

Theorem 8. *If $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ is a DLIN group then $(K_{\text{sse}}, P_{\text{sse}}, V_{\text{sse}}, S_{1, \text{sse}}, S_{\text{sse}}, E_{1, \text{sse}}, E_{\text{sse}}, SE_{1, \text{sse}})$ is an NIZK proof for R_{ppsat} with perfect completeness, perfect soundness, perfect knowledge extraction and composable zero-knowledge and unbounded simulation-sound extractability. The size of the CRS is $\mathcal{O}(1)$ group elements, while the NIZK proofs consist of $\mathcal{O}(n + \ell)$ group elements.*

² See the full paper for a DLIN group based strong one-time signature scheme.

Sketch of proof. On a real CRS, c_1 does not contain 1, and therefore by the perfect soundness of the NIZK proof c_w must contain a satisfiability witness w . In simulations, c_1 does contain 1, however, since the prover does not know the signing key sk he cannot create signatures on vk_{sots} of his own choosing and he cannot recycle a vk_{sots} either because he does not know the corresponding signing key sk_{sots} . Therefore, he cannot encrypt a signature in c_s , so he must still encrypt a satisfiability witness in c_w . We can then decrypt c_w and extract the witness. We refer to the full paper for details. \square

5 Constant Size Group Signatures Without Random Oracles

SECURITY DEFINITIONS. [7] define three security properties that a group signature must satisfy: anonymity, traceability and non-frameability. We refer to the full paper for formal definitions and to [7] for a discussion of why this is a strong security definition that incorporates previous security requirements found in the literature. The definition allows for separating the roles of the group manager into an issuer who can enroll members and an opener that can open signatures to see who created it.

Anonymity: Only the opener can see who created a signature. This property must hold even if the members' keys are exposed and the issuer is corrupt.

Traceability: If the issuer is honest then all signatures will be correctly opened to some member.

Non-frameability: Even if the issuer and opener are both corrupt, they still cannot create a valid signature and a convincing opening that frames an honest member that did not sign it.

A GROUP SIGNATURE SCHEME. We imagine that there is a PKI in place so we have authenticated public keys. We model this by having a public key registry reg where only user i has one-time write access to $reg[i]$, we do not attempt to keep this information secret. User i stores his secret key in $gsk[i]$, unless compromised only the user has access to this key.

Key generation: We create the group public key $gpk = (vk, pk, \Sigma)$, where vk is a verification key for the CMA-secure signature scheme, pk is a public key for the CPA-secure cryptosystem and Σ is a CRS for the simulation-sound extractable NIZK proof. The issuer's key ik is the signing key for the signature scheme, while the opener's key ok is the decryption key for the cryptosystem.

Join/Issue: The user i registers a public key vk_i for the CMA-secure signature scheme in $reg[i]$ and stores the corresponding secret key sk_i . The issuer signs it as $cert_i \leftarrow \text{Sign}_{ik}(vk_i)$. The user verifies the correctness of the signature and stores $gsk[i] = (sk_i, vk_i, cert_i)$.

Sign: To sign $m \in \{0, 1\}^*$, member i creates a strong one-time signature key pair $(vk_{\text{sots}}, sk_{\text{sots}})$. Using sk_i he signs the verification key, $s_i \leftarrow \text{Sign}_{sk_i}(vk_{\text{sots}})$. He then creates an encryption c of $(vk_i, cert_i, s_i)$ and makes a simulation-sound extractable NIZK proof π that the plaintext is correctly formed. Finally, he makes a strong one-time signature $s_{\text{sots}} \leftarrow \text{Sign}_{sk_{\text{sots}}}(m, vk_{\text{sots}}, c, \pi)$.

The group signature on m is $s = (vk_{\text{sots}}, c, \pi, s_{\text{sots}})$.

Verify: Accept if the strong one-time signature and the NIZK proof are valid.

Open: To open a valid group signature we decrypt c . We get some $(vk_*, cert_*, s_*)$ and look up the member i who registered vk_* . In case no such member exists, we set $i = \text{issuer}$. We return an opening (i, ψ) , where $\psi = (vk_*, cert_*, s_*)$.

Judge: Anybody can check whether $cert_*$ is a signature on vk_* under vk , and whether s_* is a signature on vk_{sots} under vk_* . If vk_* has been registered for user i , or no vk_* has been registered and $i = \text{issuer}$ we accept the opening.

Theorem 9. *If the DLIN assumption holds for \mathcal{G} then there exists a group signature scheme with anonymity, traceability and non-frameability and perfect correctness. All public keys contain $\mathcal{O}(1)$ group elements, openings contain $\mathcal{O}(1)$ group elements, and signatures contain $\mathcal{O}(1)$ group elements and elements from \mathbb{Z}_p .*

Sketch of proof. We get anonymity, because the information $(vk_i, cert_i, s_i)$ that could identify the signer is encrypted and the NIZK proof is zero-knowledge. Seeing openings of other group signatures does not help, because when a CPA-secure cryptosystem is combined with a simulation-sound proof of knowledge of the plaintext, then it becomes CCA2-secure, see also [23].

We get traceability because by the soundness of the NIZK proof system we must have a correct $(vk_*, cert_*, s_*)$ inside the ciphertext. Since only the issuer knows the signing key ik , nobody else can forge a certificate $cert_*$. This means, the group signature must point to some member i , not the issuer.

We have non-frameability because a valid signature and a valid opening pointing to i contains a signature s_* under vk_i on vk_{sots} , so vk_{sots} must have been signed by the member. Furthermore, since it is a strong one-time signature scheme and the public key vk_{sots} is used only once by i , it must also be this member that made the signature s_{sots} on $(m, vk_{\text{sots}}, c, \pi)$.

The full paper [28] contains a more detailed construction and the full proof. \square

Acknowledgment

We would like to thank Rafail Ostrovsky, Amit Sahai and Brent Waters for many discussions.

References

1. Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. <http://eprint.iacr.org/2005/385>.
2. Giuseppe Ateniese and Breno de Medeiros. Efficient group signatures without trapdoors. In *proceedings of ASIACRYPT '03, LNCS series, volume 2894*, pages 246–268, 2003. Revised paper available at <http://eprint.iacr.org/2002/173>.
3. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure group signature scheme. In *proceedings of CRYPTO '00, LNCS series, volume 1880*, pages 255–270, 2000.

4. Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 171–188, 2004. Full paper available at <http://eprint.iacr.org/2003/077>.
5. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 614–629, 2003.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pages 62–73, 1993.
7. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *proceedings of CT-RSA '05, LNCS series, volume 3376*, pages 136–153, 2005.
8. Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal of Computation*, 20(6):1084–1118, 1991.
9. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *proceedings of STOC '88*, pages 103–112, 1988.
10. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *proceedings of CRYPTO '04, LNCS series, volume 3152*, pages 41–55, 2004.
11. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *proceedings of TCC '05, LNCS series, volume 3378*, pages 325–341, 2005.
12. Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *proceedings of EUROCRYPT '06, LNCS series, volume 4004*, pages 427–444, 2006.
13. Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *proceedings of SCN '04, LNCS series, volume 3352*, pages 120–133, 2004. Full paper available at <http://www.brics.dk/~jg/GroupSignFull.pdf>.
14. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *proceedings of CRYPTO '02, LNCS series, volume 2442*, pages 61–76, 2002.
15. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *proceedings of CRYPTO '04, LNCS series, volume 3152*, pages 56–72, 2004.
16. Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In *proceedings of ASIACRYPT '98, LNCS series, volume 1514*, pages 160–174, 1998.
17. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *proceedings of CRYPTO '97, LNCS series, volume 1294*, pages 410–424, 1997.
18. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *proceedings of STOC '98*, pages 209–218, 1998.
19. Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes, 2004.
20. Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *proceedings of CRYPTO '03, LNCS series, volume 2729*, pages 565–582, 2003. Full paper available at <http://eprint.iacr.org/2003/174>.
21. David Chaum and Eugène van Heyst. Group signatures. In *proceedings of EUROCRYPT '91, LNCS series, volume 547*, pages 257–265, 1991.
22. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 566–598, 2002.
23. Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *proceedings of FOCS '92*, pages 427–436, 1992.

24. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000.
25. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.
26. Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In *proceedings of ACISP '05, LNCS series, volume 3574*, pages 455–467, 2005.
27. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *proceedings of FOCS '03*, pages 102–, 2003. Full paper available at <http://eprint.iacr.org/2003/034>.
28. Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures, 2006. Full paper available at <http://www.brics.dk/~jg/NIZKGroupSignFull.pdf>.
29. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *proceedings of CRYPTO '06, LNCS series, volume 4117*, pages 97–111, 2006.
30. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero-knowledge for np. In *proceedings of EUROCRYPT '06, LNCS series, volume 4004*, pages 339–358, 2006.
31. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 571–589, 2004. Full paper available at <http://eprint.iacr.org/2004/007>.
32. Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *proceedings of EUROCRYPT '05, LNCS series, volume 3494*, pages 198–214, 2005. Full paper available at <http://eprint.iacr.org/345>.
33. Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.
34. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *proceedings of CRYPTO '02, LNCS series, volume 2442*, pages 111–126, 2002.
35. Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS '01*, pages 543–553, 2001.
36. Brent Waters. Efficient identity-based encryption without random oracles. In *proceedings of EUROCRYPT '05, LNCS series, volume 3494*, pages 114–127, 2005.

Analysis of One Popular Group Signature Scheme

Zhengjun Cao^{1,2}

¹ Department of Mathematics, Shanghai University, Shanghai, China 200444

² Key Lab of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

zjcamss@163.com

Abstract. The group signature scheme [1], ACJT for short, is popular. In this paper we show that it is not secure. It does not satisfy exculpability. The group manager can sign on behalf of any group member. The drawback found in the scheme shows that some inductions are not sound, though they are prevalent in some so-called security proofs.

Keywords: group signature, exculpability, anonymity.

1 Introduction

Group signatures, introduced by Chaum and Heyst [2], allow individual members to make signatures on behalf of the group. Generally, a group signature must satisfy the following properties [1]:

Unforgeability: Only group members are able to sign messages on behalf of the group.

Anonymity: Given a valid signature of some message, identifying the actual signer is computationally hard for everyone but the group manager.

Unlinkability: Deciding whether two different valid signatures were produced by the same group member is computationally hard.

Traceability: The group manager is always able to open a valid signature and identify the actual signer.

Coalition-resistance: A colluding subset of group members (even if comprised of the entire group) cannot generate a valid signature that the group manager cannot link to one of the colluding group members.

Exculpability: Neither a group member nor the group manager can sign on behalf of other group member.

Group signatures can be used to constitute a very useful primitive in many settings. It has become a hot problem to research group signatures in recent [3–7].

At Crypto'2000, Ateniese et al. [1] proposed a group signature scheme. The authors claimed that the scheme was practical and provably secure coalition-resistant. Recently, we find it is false. The group manager can sign on behalf of any group member. That is to say, the popular group signature scheme does

not satisfy exculpability. It's the first time to show that the signature scheme is not secure. The attack developed in the paper is novel and interesting. The drawback found in the popular signature scheme shows that some inductions are not sound, though they are prevalent in so-called security proofs.

The rest of the paper is organized as follows. The next section reviews ACJT group signature scheme. An attack is presented in Section 3. Some conclusion remarks are given in Section 4.

2 Review

Let $\epsilon > 1, k, \ell_p$ be security parameters and let $\lambda_1, \lambda_2, \gamma_1, \gamma_2$ denote the lengths satisfying

$$\lambda_1 > \epsilon(\lambda_2 + k) + 2, \quad \lambda_2 > 4\ell_p, \quad \gamma_1 > \epsilon(\gamma_2 + k) + 2, \quad \gamma_2 > \lambda_1 + 2.$$

Define the integral ranges

$$\Lambda =] 2^{\lambda_1} - 2^{\lambda_2}, 2^{\lambda_1} + 2^{\lambda_2} [, \quad \Gamma =] 2^{\gamma_1} - 2^{\gamma_2}, 2^{\gamma_1} + 2^{\gamma_2} [.$$

Finally, let \mathcal{H} be a collision-resistant hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^k$.

The initial phase involves the group manager (GM) setting the group public key \mathcal{Y} and his secret key \mathcal{S} .

SETUP:

1. Select random secret ℓ_p -bit primes p', q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are primes. Set the modulus $n = pq$.
2. Choose random elements $a, a_0, g, h \in_R QR(n)$ (of order $p'q'$).
3. Choose a random secret element $x \in_R Z_{p'q'}^*$ and set $y = g^x \pmod n$.
4. The group public key is : $\mathcal{Y} = (n, a, a_0, y, g, h)$.
5. The corresponding secret key (known only to GM) is: $\mathcal{S} = (p', q', x)$.

Suppose now that a new user wants to join the group. We assume that communication between the user and the group manager is secure. The selection of per-user parameters is done as follows:

JOIN:

1. User P_i generates a secret exponent $\bar{x}_i \in_R]0, 2^{\lambda_2}[$, a random integer $\bar{r} \in_R]0, n^2[$ and sends $C_1 = g^{\bar{x}_i} h^{\bar{r}} \pmod n$ to GM and proves him knowledge of the representation of C_1 w.r.t. bases g and h .
2. GM checks that $C_1 \in QR(n)$. If this is the case, GM selects α_i and $\beta_i \in_R]0, 2^{\lambda_2}[$ at random and sends (α_i, β_i) to P_i .
3. User P_i computes $x_i = 2^{\lambda_1} + (\alpha_i \bar{x}_i + \beta_i \pmod{2^{\lambda_2}})$ and sends GM the value $C_2 = a^{x_i} \pmod n$. The user also proves to GM:
 - (a) that the discrete log of C_2 w.r.t. base a lies in Λ , and

- (b) knowledge of integers u, v , and ω such that
 - i. u lies in $] - 2^{\lambda_2}, 2^{\lambda_2}[$,
 - ii. u equals the discrete log of $C_2/a^{2^{\lambda_1}}$ w.r.t. base a , and
 - iii. $C_1^{\alpha_i} g^{\beta_i}$ equals $g^u (g^{2^{\lambda_2}})^v h^\omega$.

(The statements (i–iii) prove that the user’s membership secret $x_i = \log_a C_2$ is correctly computed from C_1, α_i , and β_i .)

4. GM checks that $C_2 \in QR(n)$. If this is the case and all the above proofs were correct, GM selects a random prime $e_i \in_R \Gamma$ and computes $A_i := (C_2 a_0)^{1/e_i} \bmod n$. Finally, GM sends P_i the new membership certificate $[A_i, e_i]$. (Note that $A_i = (a^{x_i} a_0)^{1/e_i} \bmod n$.)

5. User P_i verifies that $a^{x_i} a_0 \equiv A_i^{e_i} \bmod n$.

Armed with a membership certificate $[A_i, e_i]$, a group member can generate anonymous and unlinkable group signatures on a generic message $m \in \{0, 1\}^*$:

SIGN:

1. Generate a random value $\omega \in_R \{0, 1\}^{2\ell_p}$ and compute:

$$T_1 = A_i y^\omega \bmod n, \quad T_2 = g^\omega \bmod n, \quad T_3 = g^{e_i} h^\omega \bmod n.$$

2. Randomly choose $r_1 \in_R \pm\{0, 1\}^{\epsilon(\gamma_2+k)}$, $r_2 \in_R \pm\{0, 1\}^{\epsilon(\lambda_2+k)}$, $r_3 \in_R \pm\{0, 1\}^{\epsilon(\gamma_1+2\ell_p+k+1)}$, $r_4 \in_R \pm\{0, 1\}^{\epsilon(2\ell_p+k)}$ and compute:

$$d_1 = T_1^{r_1} / (a^{r_2} y^{r_3}) \bmod n, \quad d_2 = T_2^{r_1} / g^{r_3} \bmod n$$

$$d_3 = g^{r_4} \bmod n, \quad d_4 = g^{r_1} h^{r_4} \bmod n$$

$$c = \mathcal{H}(g \parallel h \parallel y \parallel a_0 \parallel a \parallel T_1 \parallel T_2 \parallel T_3 \parallel d_1 \parallel d_2 \parallel d_3 \parallel d_4 \parallel m)$$

$$s_1 = r_1 - c(e_i - 2^{\gamma_1}), \quad s_2 = r_2 - c(x_i - 2^{\lambda_1}),$$

$$s_3 = r_3 - c e_i \omega, \quad s_4 = r_4 - c \omega \quad (\text{all in } \mathbf{Z}).$$

3. Output $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$.

A verifier can check the validity of a signature $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$ on the message m as follows:

VERIFY:

1. Compute

$$c' = \mathcal{H}(g \parallel h \parallel y \parallel a_0 \parallel a \parallel T_1 \parallel T_2 \parallel T_3 \parallel d'_1 \parallel d'_2 \parallel d'_3 \parallel d'_4 \parallel m)$$

where

$$d'_1 = a_0^c T_1^{s_1 - c 2^{\gamma_1}} / (a^{s_2 - c 2^{\lambda_1}} y^{s_3}) \bmod n, \quad d'_2 = T_2^{s_1 - c 2^{\gamma_1}} / g^{s_3} \bmod n,$$

$$d'_3 = T_2^c g^{s_4} \bmod n, \quad d'_4 = T_3^c g^{s_1 - c 2^{\gamma_1}} h^{s_4} \bmod n$$

2. Accept the signature if and only if $c = c'$ and

$$s_1 \in \pm\{0, 1\}^{\epsilon(\gamma_2+k)+1}, \quad s_2 \in \pm\{0, 1\}^{\epsilon(\lambda_2+k)+1},$$

$$\underline{s_3 \in \pm\{0, 1\}^{\epsilon(\gamma_1+2\ell_p+k+1)+1}}, \quad s_4 \in \pm\{0, 1\}^{\epsilon(2\ell_p+k)+1}.$$

In case of a dispute, GM executes the following procedure:

OPEN:

1. Check the signature's validity via the VERIFY procedure.
2. Recover A_i (and thus the identity of P_i) as $A_i = T_1/T_2^x \pmod n$.
3. Prove that $\log_g y = \log_{T_2}(T_1/A_i \pmod n)$.

Remark 1: In the original description [1], we observe that

$$r_3 \in_R \pm\{0, 1\}^{\epsilon(\gamma_1+2\ell_p+k+1)}, \quad s_3 \in \pm\{0, 1\}^{\epsilon(\lambda_1+2\ell_p+k+1)+1}$$

It's not difficult to find it should be corrected to keep the consistency between r_3 and s_3 .

3 Analysis

In this section, we show that ACJT group signature scheme doesn't satisfy exculpability. More precisely, we find the group manager (GM) can sign on behalf of any member if GM replaces Step 2 in the original SETUP phase with following:

2. Choose random elements $a_0, g, h \in_R QR(n)$ (of order $p'q'$) and set $a = a_0^t \pmod n$, where $t \in_R \mathbf{Z}_{p'q'}^*$.

Then GM records (a^{x_i}, A_i, e_i) in the JOIN phase (pointing to the member P_i).

Note that no member can prevent GM from setting $a = a_0^t \pmod n$.

Using (t, a^{x_i}, A_i, e_i) and the secret key (p', q') , GM can sign on behalf of the member P_i . Given a message m , GM proceeds as follows:

1. Choose $\omega \in_R \{0, 1\}^{2\ell_p}$ and compute:

$$T_1 = A_i y^\omega \pmod n, \quad T_2 = g^\omega \pmod n, \quad T_3 = h^\omega \pmod n.$$

2. Choose $b_1, b_2 \in_R \mathbf{Z}_n, r_4 \in_R \pm\{0, 1\}^{\epsilon(2\ell_p+k)}$ and compute

$$d_1 = (a^{x_i})^{b_1} y^{b_2}, \quad d_2 = g^{b_2}, \quad d_3 = g^{r_4}, \quad d_4 = g^{b_1 e_i} h^{r_4} \pmod n.$$

$$c = \mathcal{H}(g \parallel h \parallel y \parallel a_0 \parallel a \parallel T_1 \parallel T_2 \parallel T_3 \parallel d_1 \parallel d_2 \parallel d_3 \parallel d_4 \parallel m)$$

3. Choose $X \in_R \Lambda$ and compute

$$R_1 = (c + b_1) e_i, \quad R_2 = cX + t^{-1}(c + b_1), \quad R_3 = \omega R_1 - b_2 \pmod{\phi(n)}$$

4. Choose proper $\rho_1, \rho_2, \rho_3 \in \mathbf{Z}$ such that

$$r_1 = R_1 + \rho_1 \phi(n) \in \pm\{0, 1\}^{\epsilon(\gamma_2+k)}$$

$$r_2 = R_2 + \rho_2 \phi(n) \in \pm\{0, 1\}^{\epsilon(\lambda_2+k)}$$

$$r_3 = R_3 + \rho_3 \phi(n) \in \pm\{0, 1\}^{\epsilon(\gamma_1+2\ell_p+k+1)}$$

(Since $R_1, R_2, R_3 \in \mathbf{Z}_n$, $n = (2p' + 1)(2q' + 1)$, $|p'| = |q'| = \ell_p$, $\epsilon > 1$, $\gamma_1 > \epsilon(\gamma_2 + k) + 2$, $\gamma_2 > \lambda_1 + 2$, $\lambda_1 > \epsilon(\lambda_2 + k) + 2$ and $\lambda_2 > 4\ell_p$, it's easy to find $\rho_1, \rho_2, \rho_3 \in \mathbf{Z}$ satisfying the above restrictions.)

5. Compute

$$s_1 = r_1 - c(e_i - 2^{\gamma_1}), \quad s_2 = r_2 - c(X - 2^{\lambda_1}),$$

$$s_3 = r_3 - ce_i\omega, \quad s_4 = r_4 - c\omega \quad (\text{all in } \mathbf{Z}).$$

6. Output $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$.

Correctness: For convenience, denote by ξ_i the inverse of e_i modulo $\phi(n)$, i.e.,

$$e_i \xi_i = 1 \pmod{\phi(n)}$$

Hence, we have

$$\begin{aligned} d'_1 &= a_0^c T_1^{s_1 - c2^{\gamma_1}} / (a^{s_2 - c2^{\lambda_1}} y^{s_3}) = a_0^c (A_i y^\omega)^{r_1 - ce_i} / (a^{r_2 - cX} y^{r_3 - ce_i\omega}) \\ &= a_0^c ((a^{x_i} a_0)^{\xi_i})^{(r_1 - ce_i)} y^{\omega r_1 - r_3} / a^{r_2 - cX} \\ &= (a^{x_i})^{r_1 \xi_i - c} a_0^{c + r_1 \xi_i - c} y^{\omega r_1 - r_3} / a^{r_2 - cX} = (a^{x_i})^{r_1 \xi_i - c} a_0^{r_1 \xi_i} y^{\omega r_1 - r_3} / a_0^{t(r_2 - cX)} \\ &= (a^{x_i})^{r_1 \xi_i - c} a_0^{r_1 \xi_i - t(r_2 - cX)} y^{\omega r_1 - r_3} = (a^{x_i})^{R_1 \xi_i - c} a_0^{R_1 \xi_i - t(R_2 - cX)} y^{\omega R_1 - R_3} \\ &= (a^{x_i})^{b_1} a_0^{c + b_1 - t(c + b_1)t^{-1}} y^{b_2} = (a^{x_i})^{b_1} y^{b_2} = d_1 \pmod{n} \end{aligned}$$

$$\begin{aligned} d'_2 &= T_2^{s_1 - c2^{\gamma_1}} / g^{s_3} = (g^\omega)^{r_1 - ce_i} / g^{r_3 - ce_i\omega} \\ &= g^{\omega r_1 - r_3} = g^{\omega R_1 - R_3} = g^{b_2} = d_2 \pmod{n} \end{aligned}$$

$$d'_3 = T_2^c g^{s_4} = (g^\omega)^c g^{r_4 - \omega c} = g^{r_4} = d_3 \pmod{n}$$

$$\begin{aligned} d'_4 &= T_3^c g^{s_1 - c2^{\gamma_1}} h^{s_4} = (h^\omega)^c g^{r_1 - ce_i} h^{r_4 - c\omega} \\ &= g^{R_1 - ce_i} h^{r_4} = g^{b_1 e_i} h^{r_4} = d_4 \pmod{n} \end{aligned}$$

Thus $c' = c$. It's easy to check that

$$s_1 \in \pm\{0, 1\}^{\epsilon(\gamma_2 + k) + 1}, \quad s_2 \in \pm\{0, 1\}^{\epsilon(\lambda_2 + k) + 1},$$

$$s_3 \in \pm\{0, 1\}^{\epsilon(\gamma_1 + 2\ell_p + k + 1) + 1}, \quad s_4 \in \pm\{0, 1\}^{\epsilon(2\ell_p + k) + 1}.$$

Clearly, we also have

$$T_1 / T_2^x = A_i y^\omega / (g^\omega)^x = A_i \pmod{n}$$

Therefore, the scheme is not exculpable.

Remark 2: The authors [1] claimed that

First note that due to Corollary 2, GM does not get any information about a user's secret x_i apart from a^{x_i} . Thus, the value x_i is computationally hidden from GM. Next note that T_1, T_2 , and T_3 are an unconditionally binding commitments to A_i and e_i . One can show that, if the factorization of n would be publicly known, the interactive proof underlying the group signature scheme is a proof of knowledge of the discrete log of $A_i^{e_i}/a_0$ (provided that ℓ_p is larger than twice to output length of the hash function / size of the challenges). Hence, not even the group manager can sign on behalf of P_i because computing discrete logarithms is assumed to be infeasible.

But by the above attack, GM is not forced to know a user's secret x_i even that T_1, T_2 , and T_3 are an unconditionally binding commitments to A_i and e_i . We should stress that the likes of the above induction are not sound, though they are prevalent in some so-called security proofs.

4 Conclusion

In this paper we show that ACJT group signature scheme is insecure. The attack introduced in the paper will be helpful for researching group signature schemes in the future. Incidentally, the fair E-cash system [8] directly based on ACJT fails. But it seems that the attack does not apply to the extensions of ACJT proposed in [9]. The extension proposed in [10] appears to resist the attack at the cost of the presence of a trusted third party.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Professor M.L. Liu for many enlightening suggestions. Thanks also go to those anonymous referees who contributed with their expertise to the final version of the paper. This work is supported by National Natural Science Foundation of China (90304012) and Project 973 (2004CB318000).

References

- [1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. *CRYPTO'2000*, LNCS 1880, pp.255–270. Springer-Verlag, 2000.
- [2] D. Chaum and E. van Heyst. Group signatures. *EUROCRYPT'1991*, LNCS 547, pp.257–265. Springer-Verlag, 1992.
- [3] D. Song. Practical forward-secure group signature schemes. *ACM Symposium on Computer and Communication Security*, pp.225–234, November 2001.
- [4] E. Bresson and J. Stern. Efficient revocation in group signatures. *PKC'2001*, LNCS 1992, pp.190–206. Springer-Verlag, 2001.
- [5] G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. *ASIACRYPT'2003*, LNCS 2894, pp.246–268. Springer-Verlag, 2003.

- [6] Mihir Bellare, Daniele Micciancio, Bogdan Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. *EUROCRYPT'2003*, LNCS 2656, pp.614–629. Springer-Verlag, 2003.
- [7] D. Boneh, X. Boyen and H. Shacham. Short group signatures. *CRYPTO'2004*, LNCS 3152, pp.41–55. Springer-Verlag, 2004.
- [8] Greg Maitland and Colin Boyd. Fair electronic cash based on a group signature scheme. *Information and Communications Security'2001*, LNCS 2229, pp.461–465. Springer-Verlag, 2001.
- [9] J. Camenisch and J. Groth. Group Signatures: Better Efficiency and New Theoretical Aspects: *Forth Conference on Security in Communication Networks-SCN'04*. LNCS 3352, pp.120–133. Springer-Verlag, 2005.
- [10] G. Tsudik, S. Xu. Accumulating composites and improved group signing. *ASIACRYPT 2003*, LNCS 2894, pp.269–286. Springer-Verlag, 2003.

Author Index

- Abdalla, Michel 332
Araki, Toshinori 364
Attrapadung, Nuttapon 161
Avanzi, Roberto 130
- Baignères, Thomas 380
Bellare, Mihir 299
Berbain, Côme 396
Biham, Eli 412
Boldyreva, Alexandra 210
- Cao, Zhengjun 460
Chang, Donghoon 283
Chatterjee, Sanjit 145
Contini, Scott 37
- De Cannière, Christophe 1
Dimitrov, Vassil 130
Doche, Christophe 130
Dunkelman, Orr 412
- Feng, Dengguo 54
Finiasz, Matthieu 380
Fischlin, Marc 210
Furukawa, Jun 161
- Galindo, David 178
Gaudry, P. 114
Gilbert, Henri 315
Groth, Jens 444
- Herranz, Javier 178
Houtmann, T. 114
- Imai, Hideki 161
- Jochemsz, Ellen 267
- Keller, Nathan 412
Kiltz, Eike 178
Kohel, D. 114
- Kunihiro, Noboru 21
Kurosawa, Kaoru 428
- Leander, Gregor 241
Lee, Sangjin 283
Li, Na 84
Li, Qiming 99
- May, Alexander 267
Memon, Nasir 99
Muller, Frédéric 315
- Nachef, Valérie 396
Naito, Yusuke 21
Nandi, Mridul 283
Nikov, Ventsislav 348
Nikova, Svetla 348
- Obana, Satoshi 364
Ogata, Wakaha 226
Ohta, Kazuo 21
- Paillier, Pascal 252
Patarin, Jacques 396
Paul, Souradyuti 69
Peyrin, Thomas 315
Pieprzyk, Josef 194
Pointcheval, David 332
Preneel, Bart 69, 348
- Qi, Wen-Feng 84
- Rechberger, Christian 1
Ristenpart, Thomas 299
Ritzenthaler, C. 114
Robshaw, Matt 315
Rupp, Andy 241
- Sarkar, Palash 145
Sasaki, Yu 21
Shimoyama, Takeshi 21
Sica, Francesco 130

Steinfeld, Ron 194
Sutcu, Yagiz 99

Takagi, Tsuyoshi 428
Teranishi, Isamu 226

Villar, Jorge L. 252

Wang, Huaxiong 194
Weng, A. 114

Yajima, Jun 21
Yin, Yiqun Lisa 37
Yung, Moti 283

Zhang, Bin 54