Antoine Joux (Ed.)

LNCS 5479

Advances in Cryptology – EUROCRYPT 2009

28th Annual International Conference on the Theory and Applications of Cryptographic Techniques
Cologne, Germany, April 2009, Proceedings

Springer

Lecture Notes in Computer Science 5479

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

Antoine Joux (Ed.)

Advances in Cryptology – EUROCRYPT 2009

28th Annual International Conference on the Theory and Applications of Cryptographic Techniques
Cologne, Germany, April 26-30, 2009
Proceedings

Springer

Volume Editor

Antoine Joux
DGA and University of Versailles Saint-Quentin-en-Yvelines
45, avenue des Etats-Unis, 78035 Versailles Cedex, France
E-mail: antoine.joux@m4x.org

Library of Congress Control Number: Applied for

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-642-01000-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-01000-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12648559 06/3180 5 4 3 2 1 0

The original version of the book was revised: The copyright line was incorrect. The Erratum to the book is available at
DOI: 10.1007/978-3-642-01001-9_35

Preface

You are holding the proceedings of Eurocrypt 2009, the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques. This conference was organized by the International Association for Cryptologic Research in cooperation with the Horst Görtz Institute for IT-Security at the Ruhr-Universität Bochum. The local organization received additional support from several sponsors: Horst Görtz Stiftung, Deutsche Forschungsgemeinschaft, Bochum 2015, Secunet, NXP, IET, Taylor & Francis, AuthentiDate. The conference was held in Cologne, Germany.

The Eurocrypt 2009 Program Committee (PC) consisted of 29 members, listed on the next page. There were 148 submissions and 33 were selected to appear in this volume. Each submission was assigned to at least three PC members and reviewed anonymously. During the review process, the PC members were assisted by 131 external reviewers. Once the reviews were available, the committee discussed the papers in depth using the EasyChair conference management system. The authors of accepted papers were given five weeks to prepare the final versions included in these proceedings. The revised papers were not reviewed again and their authors bear the responsibility for their content.

In addition to the papers included in this volume, the conference also featured a Poster and a Rump session. The list of presented posters appears in this volume before the table of contents. Dan Bernstein served as the Chair of the Rump session. The conference also had the pleasure of hearing invited talks by Shafi Goldwasser and Phillip Rogaway.

The PC decided to give the Best Paper Award to Dennis Hofheinz and Eike Kiltz for their paper "Practical Chosen Ciphertext Secure Encryption from Factoring." In addition, the PC selected two other papers for invitation to the *Journal of Cryptology*: "On Randomizing Some Hash Functions to Strengthen the Security of Digital Signatures" by Praveen Gauravaram and Lars Knudsen, and "Possibility and Impossibility Results for Encryption and Commitment Secure Under Selective Opening" by Mihir Bellare, Dennis Hofheinz and Scott Yilek.

I wish to thank all the people who contributed to this conference. First, all the authors who submitted their work. The PC members and their external reviewers for the thorough job they did while reading and commenting on the submissions. Without them, selecting the papers for this conference would have been an impossible task. I thank Andrei Voronkov for his review system EasyChair, I was especially impressed by the tools that helped me while assembling this volume. I am grateful to Arjen Lenstra for the help and advice he gave as representative of the IACR Board. I also would like to thank the General Chair Alexander May and his Co-chairs for making this conference possible.

Being the Program Chair for Eurocrypt 2009 was a great honor and I may only hope that the readers of these proceedings will find them as interesting as I found the task of selecting their content.

February 2009 Antoine Joux

Organization

General Chair

Alexander May	Ruhr-Universität Bochum, Germany

Co-chairs

Roberto Avanzi	Christof Paar	Ahmad Sadeghi
Jörg Schwenk	Christopher Wolf	

Program Chair

Antoine Joux	DGA and Université de Versailles Saint-Quentin-en-Yvelines, France

Program Committee

Paulo Barreto	University of São Paulo, Brazil
Alexandra Boldyreva	Georgia Institute of Technology, USA
Colin Boyd	Queensland University of Technology, Australia
Xavier Boyen	Stanford University, USA
Mike Burmester	Florida State University, USA
Serge Fehr	CWI Amsterdam, The Netherlands
Marc Fischlin	TU Darmstadt, Germany
Pierre-Alain Fouque	École Normale Supérieure, Paris, France
Craig Gentry	Stanford University, USA
Henri Gilbert	Orange Labs, France (Eurocrypt 2010 Chair)
Helena Handschuh	Spansion, France
Nick Howgrave-Graham	NTRU Cryptosystems, USA
Thomas Johansson	Lund University, Sweden
Jonathan Katz	University of Maryland and IBM Research, USA
John Kelsey	National Institute of Standards and Technology, USA
Kwangjo Kim	Information and Communications University, Korea
Kaoru Kurosawa	Ibaraki University, Japan
Reynald Lercier	DGA/CELAR and Université de Rennes, France
Anna Lysyanskaya	Brown University, USA
Rafail Ostrovsky	University of California, Los Angeles, USA

Pascal Paillier	Gemalto Security Labs/Crytography & Innovation, France
Duong Hieu Phan	Université de Paris 8, France
Christian Rechberger	IAIK, Graz University of Technology, Austria
Werner Schindler	Bundesamt für Sicherheit in der Informationstechnik, Germany
Thomas Shrimpton	Portland State University and University of Lugano, USA and Italy
Nigel Smart	University of Bristol, UK (Eurocrypt 2008 Chair)
Rainer Steinwandt	Florida Atlantic University, USA
Christine Swart	University of Cape Town, South Africa
Christopher Wolf	Ruhr University Bochum, Germany

External Reviewers

Abdalla, Michel
Abe, Masayuki
Andreeva, Elena
Armknecht, Frederik
Bangerter, Endre
Bellare, Mihir
Benaloh, Josh
Bernstein, Daniel J.
Billet, Olivier
Bouillaguet, Charles
Broker, Reinier
Brown, Dan
Cash, David
Chandran, Nishanth
Chen, Lidong
Chevallier-Mames, Benoît
Clavier, Christophe
Cochran, Martin
Coron, Jean-Sébastien
Dent, Alex
Dodis, Yevgeniy
Duc, Dang Nguyen
Fiore, Dario
Fischer, Jan
Furukawa, Jun
Galbraith, Steven D.
Garay, Juan
Gazzoni Filho, Décio Luiz
Gebhardt, Max
Gennaro, Rosario
Gonzalez, Juan
Goubin, Louis
Gouget, Aline
Goyal, Vipul
van de Graaf, Jeroen
Halevi, Shai
Hanaoka, Goichiro
Hemenway, Brett
Heng, Swee Huay
Herbst, Christoph
Herranz, Javier
Hisil, Huseyin
Hoeper, Katrin
Hofheinz, Dennis
Holz, Thorsten
Hutter, Michael
Iorga, Michaela
Ishai, Yuval
Iwata, Tetsu
Jacobson, Michael
Jain, Abhishek
Kiltz, Eike
Koshiba, Takeshi
Krawczyk, Hugo
Kursawe, Klaus
Lamberger, Mario
Lange, Tanja
Lee, Younho

Lehmann, Anja
Lenstra, Arjen
Lindell, Yehuda
Lochter, Manfred
Lu, Steve
Lucks, Stefan
Lyubashevsky, Vadim
Margraf, Marian
Maximov, Alexander
Mendel, Florian
Montenegro, Jose
Moran, Tal
Morrissey, Paul
Moss, Andrew
Naccache, David
Nad, Tomislav
Naehrig, Michael
Namprempre, Chanathip
Neven, Gregory
Nguyen, Phong
Niedermeyer, Frank
Noack, Andreas
O'Neill, Adam
Ogata, Wakaha
Ohkubo, Miyako
Oliveira, Leonardo
Oswald, Elisabeth
Page, Dan
Pandey, Omkant
Paul, Souradyuti
Peikert, Chris
Perlner, Ray
Persiano, Giuseppe
Pietrzak, Krzysztof
Pointcheval, David
Poschmann, Axel
Preneel, Bart
Priemuth-Schmid, Deike
Quisquater, Jean-Jacques
Ramzan, Zulfikar
Rappe, Dörte
Regenscheid, Andrew
Rezaeian Farashahi, Reza
Ristenpart, Thomas
Rose, Greg
Sakane, Hirofumi
Schläffer, Martin
Schmidt, Jörn-Marc
Schoenmakers, Berry
Schröder, Dominique
Schulte-Geers, Ernst
Segev, Gil
Shacham, Hovav
Shparlinski, Igor
Spitz, Stefan
Stam, Martijn
Stein, Oliver
Steinberger, John
Szekely, Alexander
Tillich, Stefan
Toft, Tomas
Tuengerthal, Max
Tunstall, Michael
Van Assche, Gilles
Vercauteren, Frederik
Vergnaud, Damien
Visconti, Ivan
Warinschi, Bogdan
Waters, Brent
Wee, Hoeteck
Wolf, Stefan
Wyseur, Brecht
Yerukhimovich, Arkady
Zenner, Erik
Zimmer, Sébastien

List of Presented Posters

Physically Unclonable Pseudorandom Functions
Frederik Armknecht, Ahmad-Reza Sadeghi, Pim Tuyls, Roel Maes and Berk Sunar

Automatic Generation of sound Zero-Knowledge Protocols
Endre Bangerter, Jan Camenisch, Stephan Krenn, Ahmad-Reza Sadeghi and Thomas Schneider

On the Data Complexity of Statistical Attacks Against Block Ciphers
Céline Blondeau and Benoît Gérard

Anonymity from Asymmetry: New Constructions for Anonymous HIBE
Dan Boneh and Léo Ducas

Pairing with Supersingular Trace Zero Varieties Revisited
Emanuele Cesena

Odd-Char Multivariate Hidden Field Equations
Ming-Shing Chen, Jintai Ding, Chia-Hsin Owen Chen, Fabian Werner and Bo-Yin Yang

Finding Good Linear Approximations of Block Ciphers and its Application to Cryptanalysis of Reduced Round DES
Rafaël Fourquet, Pierre Loidreau and Cédric Tavernier

Techniques for Public Key Cryptographic Acceleration on Graphics Processors
Owen Harrison and John Waldron

Statistical Tests for Key Recovery Using Multidimensional Extension of Matsui's Algorithm 1
Miia Hermelin, Joo Yeon Cho and Kaisa Nyberg

The Key-Dependent Attack on Block Ciphers
Xiaorui Sun and Xuejia Lai

On Privacy Losses in the Trusted Agent Model
Paulo Mateus and Serge Vaudenay

Solving Low-Complexity Ciphers with Optimized SAT Solvers
Karsten Nohl and Mate Soos

A Geometric Approach on Pairings and Hierarchical Predicate Encryption.
Tatsuaki Okamoto and Katsuyuki Takashima

Table of Contents

Security, Proofs and Models (1)

Hash Cryptanalysis

Group and Broadcast Encryption

Cryptosystems (1)

Cryptanalysis

Cryptosystems (2)

Invited Talk

Security, Proofs and Models (2)

Side Channels

Curves

Randomness

Possibility and Impossibility Results for Encryption and Commitment Secure under Selective Opening

Mihir Bellare[1], Dennis Hofheinz[2], and Scott Yilek[1]

[1] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA
{mihir,syilek}@cs.ucsd.edu
http://www-cse.ucsd.edu/users/{mihir,syilek}
[2] CWI, Amsterdam
Dennis.Hofheinz@cwi.nl
http://www.cwi.nl/~hofheinz

Abstract. The existence of encryption and commitment schemes secure under selective opening attack (SOA) has remained open despite considerable interest and attention. We provide the first public key encryption schemes secure against sender corruptions in this setting. The underlying tool is lossy encryption. We then show that no non-interactive or perfectly binding commitment schemes can be proven secure with black-box reductions to standard computational assumptions, but any statistically hiding commitment scheme is secure. Our work thus shows that the situation for encryption schemes is very different from the one for commitment schemes.

1 Introduction

IND-CPA and IND-CCA are generally viewed as strong notions of encryption security that suffice for applications. However, there is an important setting where these standard notions do not in fact imply security and the search for solutions continues, namely, in the presence of selective-opening attack (SOA) [22, 13, 38, 18, 16, 14]. Let us provide some background on SOA and then discuss our results for encryption and commitment.

1.1 Background

THE PROBLEM. Suppose a receiver with public encryption key pk receives a vector $\mathbf{c} = (\mathbf{c}[1], \ldots, \mathbf{c}[n])$ of ciphertexts, where sender i created ciphertext $\mathbf{c}[i] = \mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$ by encrypting a message $\mathbf{m}[i]$ under pk and coins $\mathbf{r}[i]$ $(1 \leq i \leq n)$. It is important here that *the messages* $\mathbf{m}[1], \ldots, \mathbf{m}[n]$ *might be related*, but *the coins* $\mathbf{r}[1], \ldots, \mathbf{r}[n]$ *are random and independent*. Now, the adversary, given $\mathbf{c}$, is allowed to corrupt some size t subset $I \subseteq \{1, \ldots, n\}$ of senders (say $t = n/2$), obtaining not only their messages but *also their coins*, so that

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 1–35, 2009.

it has $\mathbf{m}[i], \mathbf{r}[i]$ for all $i \in I$. This is called a selective opening attack (SOA). The security requirement is that the privacy of the unopened messages, namely $\mathbf{m}[i_1], \ldots, \mathbf{m}[i_{n-t}]$ where $\{i_1, \ldots, i_{n-t}\} = \{1, \ldots, n\} \setminus I$, is preserved. (Meaning the adversary learns nothing more about the unopened messages than it could predict given the opened messages and knowledge of the message distribution. Formal definitions to capture this will be discussed later.) The question is whether SOA-secure encryption schemes exist.

Status and motivation. One's first impression would be that a simple hybrid argument would show that any IND-CPA scheme is SOA-secure. Nobody has yet been able to push such an argument through. (And, today, regarding whether IND-CPA implies SOA-security we have neither a proof nor a counterexample.) Next one might think that IND-CCA, at least, would suffice, but even this is not known. The difficulty of the problem is well understood and documented [22, 13, 16, 38, 18, 14], and whether or not SOA-secure schemes exist remains open.

Very roughly, the difficulties come from a combination of two factors. The first is that it is the random coins underlying the encryption, not just the messages, that are revealed. The second is that the messages can be related.

We clarify that the problem becomes moot if senders can erase their randomness after encryption, but it is well understood that true and reliable erasure is difficult on a real system. We will only be interested in solutions that avoid erasures.

The problem first arose in the context of multiparty computation, where it is standard to assume secure communication channels between parties [8, 17]. But, how are these to be implemented? Presumably, via encryption. But due to the fact that parties can be corrupted, the encryption would need to be SOA-secure. We contend, however, that there are important practical motivations as well. For example, suppose a server has SSL connections with a large number of clients. Suppose a virus corrupts some fraction of the clients, thereby exposing the randomness underlying their encryptions. Are the encryptions of the uncorrupted clients secure?

Commitment. Notice that possession of the coins allows the adversary to verify that the opening is correct, since it can compute $\mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$ and check that this equals $\mathbf{c}[i]$ for all $i \in I$. This apparent commitment property has been viewed as the core technical difficulty in obtaining a proof. The view that commitment is in this way at the heart of the problem has led researchers to formulate and focus on the problem of commitment secure against SOA [22]. Here, think of the algorithm $\mathcal{E}$ in our description above as the commitment algorithm of a commitment scheme, with the public key being the empty string. The question is then exactly the same. More generally the commitment scheme could be interactive or have a setup phase.

Independently of the encryption setting, selective openings of commitments commonly arise in zero-knowledge proofs. Namely, often an honest verifier may request that the prover opens a subset of a number of previously made commitments. Thus, SOA-security naturally becomes an issue here, particularly when considering the concurrent composition of zero-knowledge proofs (since then,

overall more openings from a larger set of commitments may be requested). The security of the unopened commitments is crucial for the zero-knowledge property of such a protocol, and this is exactly what SOA-security of the commitments would guarantee.

Definitions. Previous work [22] has introduced and used a semantic-style security formalization of security under SOA. A contribution of our paper is to provide an alternative indistinguishability-based formalization that we denote IND-SO-ENC for encryption and IND-SO-COM for commitment. We will also refer to semantic security formalizations SEM-SO-ENC and SEM-SO-COM.

1.2 Results for Encryption

We provide the first public-key encryption schemes provably secure against selective-opening attack. The schemes have short keys. (Public and secret keys of a fixed length suffice for encrypting an arbitrary number of messages.) The schemes are stateless and noninteractive, and security does not rely on erasures. The schemes are without random oracles, proven secure under standard assumptions, and even efficient. We are able to meet both the indistinguishability (IND-SO-ENC) and the semantic security (SEM-SO-ENC) definitions, although under different assumptions.

Closer look. The main tool (that we define and employ) is lossy encryption, an encryption analogue of lossy trapdoor functions [40] that is closely related to meaningful-meaningless encryption [34] and dual-mode encryption [41]. We provide an efficient implementation of lossy encryption based on DDH. We also show that any (sufficiently) lossy trapdoor function yields lossy encryption, thereby obtaining several other lossy encryption schemes via the lossy trapdoor constructions of [40, 10, 45].

We then show that any lossy encryption scheme is IND-SO-ENC secure, thereby obtaining numerous IND-SO-ENC secure schemes. If the lossy encryption scheme has an additional property that we call efficient openability, we show that it is also SEM-SO-ENC secure. We observe that the classical quadratic residuosity-based encryption scheme of Goldwasser and Micali [27] is lossy with efficient openability, thereby obtaining SEM-SO-ENC secure encryption. It is interesting in this regard that the solution to a long-standing open problem is a scheme that has been known for 25 years. (Only the proof was missing until now.)

Previous work. In the version of the problem that we consider, there is one receiver and many senders. Senders may be corrupted, with the corruption exposing their randomness and message. An alternative version of the problem considers a single sender and many receivers, each receiver having its own public and secret key. Receivers may be corrupted, with corruption exposing their secret key. Previous work has mostly focused on the receiver corruption version of the problem. Canetti, Feige, Goldreich and Naor [13] introduce and implement non-committing encryption, which yields SOA-secure encryption in the receiver corruption setting. However, their scheme does not have short keys. (Both the

public and the secret key in their scheme are as long as the total number of message bits ever encrypted.) Furthermore, Nielsen [38] shows that this is necessary. Canetti, Halevi and Katz [16] provide SOA-secure encryption schemes for the receiver corruption setting with short public keys, but they make use of (limited) erasures. (They use a key-evolving system where, at the end of every day, the receiver's key is updated and the previous version of the key is securely erased.) In the symmetric setting, Panjwani [39] proves SOA-security against a limited class of attacks.

Our schemes do not suffer from any of the restrictions of previous ones. We have short public and secret keys, do not rely on erasures, and achieve strong notions of security.

A natural question is why our results do not contradict Nielsen's negative result saying that no noninteractive public key encryption scheme with short and fixed keys is SOA-secure without erasures for an unbounded number of messages [38]. The reason is that we consider sender corruptions as opposed to receiver corruptions.

Discussion. It has generally been thought that the two versions of the problem (sender or receiver corruptions) are of equal difficulty. The reason is that corruptions, in either case, allow the adversary to verify an opening and appear to create a commitment. (Either the randomness or the decryption key suffices to verify an opening.) Our work refutes this impression and shows that sender corruptions are easier to handle than receiver ones. Indeed, we can fully resolve the problem in the former case, while the latter case remains open. (Achieving a simulation-based notion for receiver corruptions is ruled out by [38] but achieving an indistinguishability-based notion may still be possible.)

1.3 Results for Commitment

Previous work. In the zero-knowledge (ZK) setting, Gennaro and Micali [24] notice a selective opening attack and circumvent it by adapting the distribution of the messages committed to. Similarly, a number of works (e.g.,Dolev et al. [21], Prabhakaran et al.[42] in the ZK context) use "cut-and-choose" techniques on committed values, which is a specific form of selective opening. These works can prove security by using specific properties of the distributions of the committed values (e.g., the fact that the unopened values, conditioned on the opened values, are still uniformly distributed). The first explicit treatment of SOA-secure commitment is by Dwork, Naor, Reingold, and Stockmeyer [22]. They formalized the problem and defined SEM-SO-COM. On the negative side, they showed that the existence of a one-shot (this means non-interactive and without setup assumptions) SEM-SO-COM-secure commitment scheme implied solutions to other well-known cryptographic problems, namely, three-round ZK and "magic functions." This is evidence that simulation-based one-shot SOA-secure commitment is difficult to achieve. In particular, from Goldreich and Krawczyk [26], it is known that three-round black-box zero-knowledge proof systems exist only for

languages in BPP.[1] On the positive side Dwork et al. showed that any statistically hiding chameleon commitment scheme is SOA-secure. (This scheme would not be one-shot, which is why this does not contradict their negative results.)

Results for SEM-SO-COM. On the negative side, we show that no one-shot or perfectly binding commitment scheme can be shown SEM-SO-COM-secure using black-box reductions to standard assumptions. Here, by a standard assumption, we mean any assumption that can be captured by a game between a challenger and an adversary. (A more formal definition will be given later.) Most (but not all) assumptions are of this form. On the positive side, we show, via non-black-box techniques, that there exists an interactive SEM-SO-COM-secure commitment scheme under the assumption that one-way permutations exist.

Results for IND-SO-COM. On the negative side, we show that no perfectly hiding commitment scheme (whether interactive or not) can be shown IND-SO-COM secure using black-box reductions to standard assumptions. On the positive side, we show that any statistically hiding commitment scheme is IND-SO-COM secure. (We note that a special case of this result was already implicit in the work of Bellare and Rogaway [6].)

Closer look. Technically, we derive black-box impossibility results in the style of Impagliazzo and Rudich [32], but we can derive stronger claims, similar to Dodis et al. [20]. (Dodis et al. [20] show that the security of full-domain hash signatures [4] cannot be proved using a black-box reduction to any hardness assumption that is satisfied by a random permutation.) Concretely, we prove impossibility of $\forall\exists$semi-black-box proofs from *any* computational assumption that can be formalized as an oracle $\mathcal{X}$ and a corresponding security property $\mathcal{P}$ (i.e., a game between a challenger and an adversary) which the oracle satisfies. For instance, to model one-way permutations, $\mathcal{X}$ could be a truly random permutation and $\mathcal{P}$ could be the one-way game in which a PPT adversary tries to invert a random image. We emphasize that, somewhat surprisingly, our impossibility claim holds even if $\mathcal{P}$ models SOA-security. In that case, however, a reduction will necessarily be non-black-box, see Section 9 for a discussion. Concurrently to and independently from our work, Haitner and Holenstein [28] developed a framework to prove impossibility of black-box reductions from *any* computational assumption. While their formalism is very similar to ours (e.g., their definition of a "cryptographic game" matches our definition of a "property"), they apply it to an entirely different problem, namely, encryption scheme security in the presence of key-dependent messages.

[1] "Black-box" means here that the ZK simulator uses only the (adverserial) verifier's next-message function in a black-box way to simulate an authentic interaction. Jumping ahead, we will show that in many cases SOA-secure commitment cannot be proved using a black-box reduction to a standard computational assumption. Both statements are negative, but orthogonal. Indeed, it is conceivable that a security reduction uses specific, non-black-box properties of the adversary (e.g., it is common in reductions to explicitly make use of the adversary's complexity bounds), but neither scheme nor reduction use specifics (like the code) of the underlying primitive.

Relation to the encryption results. An obvious question is why our results for encryption and commitment are not contradictive. The answer is that our SOA-secure encryption scheme does not give rise to a commitment scheme. Our commitment results do show that the SOA-security of an encryption scheme cannot be proved using a black-box reduction, *but only if encryption constitutes a commitment.* Because we consider SOA-security under *sender* corruptions in the encryption setting, this is not the case. (Recall that with sender corruptions, an encryption opening does not reveal the secret key, so the information-theoretic argument of Nielsen [38] that any encryption scheme is committing does not apply.)

1.4 History

This paper was formed by merging two Eurocrypt 2009 submissions which were accepted by the PC under the condition that they merge. One, by Bellare and Yilek, contained the results on encryption. (Sections 1.1,3,4,5.) The other, by Hofheinz, contained the results on commitment. (Sections 1.2,6,7,8,9.) Both papers had independently introduced the indistinguishability definition of SOA-security, the first for encryption and the second for commitment. Full versions of both papers are available as [7, 31].

2 Notation

For any integer n, let 1^n be its unary representation and let $[n]$ denote the set $\{1, \ldots, n\}$. We let $a \leftarrow b$ denote assignment to a the result of evaluating b. If b is simply a tuple of values of size m, we will write $(b_1, \ldots, b_m) \leftarrow b$ when we mean that b is parsed into b_1 to b_m. We let $a \leftarrow\!\!\$\ b$ denote choosing a value uniformly at random from random variable b and assigning it to a.

We say a function $\mu(n)$ is negligible if $\mu \in o(n^{-\omega(1)})$. We let $\mathsf{neg}(n)$ denote an arbitrary negligible function. If we say some $p(n) = \mathsf{poly}(n)$, we mean that there is some polynomial q such that for all sufficiently large n, $p(n) \leq q(n)$. The statistical distance between two random variable X and Y over common domain D is $\Delta(X, Y) = \frac{1}{2} \sum_{z \in D} |\Pr[X = z] - \Pr[Y = z]|$ and we say that two random variables X and Y are δ-close if their statistical distance is at most δ and if δ is negligible, we might say $X \equiv_s Y$.

We denote by ϵ the empty string. For any strings m_0 and m_1, let $m_0 \oplus m_1$ denote the bitwise xor of the two strings. We use boldface letters for vectors, and for any vector $\mathbf{m}$ of n messages and $i \in [n]$, let $\mathbf{m}[i]$ denote the ith message in $\mathbf{m}$. For a set $I \subseteq [n]$ of indices $i_1 < i_2 < \ldots < i_l$, let $\mathbf{m}[I] = (\mathbf{m}[i_1], \mathbf{m}[i_2], \ldots, \mathbf{m}[i_l])$. For any set I (resp. any vector $\mathbf{m}$)(resp. any string m), let $|I|$ (resp. $|\mathbf{m}|$) (resp. $|m|$) denote the size of the set (resp. length of the vector) (resp. length of the string).

All algorithms in this paper are randomized, unless otherwise specified as being deterministic. For any algorithm A, let $\mathsf{Coins}_A(x_1, x_2, \ldots)$ denote the set of possible coins A uses when run on inputs $x_1, x_2, \ldots$. Let $A(x_1, x_2, \ldots; r)$ denote running algorithm A on inputs $x_1, x_2, \ldots$ and with coins $r \in \mathsf{Coins}_A(x_1, x_2, \ldots)$. Then $A(x_1, x_2, \ldots)$ denotes the random variable $A(x_1, x_2, \ldots; r)$ with r chosen

uniformly at random from $\mathsf{Coins}_A(x_1, x_2, \ldots)$. When we say an algorithm is efficient, we mean that it runs in polynomial time in its first input; if the algorithm is randomized we might also say it runs in probabilistic polynomial time (PPT). An unbounded algorithm does not necessarily run in polynomial time.

3 Encryption Related Definitions

3.1 Encryption Schemes

A public-key encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of PT algorithms. The (randomized) key generation algorithm $\mathcal{K}$ takes as input a security parameter 1^λ and outputs a public key/secret key pair (pk, sk). The (randomized) encryption algorithm $\mathcal{E}$ takes as input a public key pk and a message m and outputs a ciphertext c. The decryption algorithm takes as input a secret key sk and a ciphertext C and outputs either the decryption m of c, or $\perp$, denoting failure. We require the correctness condition that for all (pk, sk) generated by $\mathcal{K}$, and for all messages m, $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$. The standard notion of security for public-key encryption scheme is indistinguishability under chosen-plaintext attack ($\mathsf{ind\text{-}cpa}$).

3.2 Encryption Security under Selective Opening

We consider both indistinguishability-based and simulation-based definitions of security for encryption under selective opening which we call $\mathsf{ind\text{-}so\text{-}enc}$ and $\mathsf{sem\text{-}so\text{-}enc}$, respectively.

Indistinguishability-based. For any public-key encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, any message sampler $\mathcal{M}$, and any adversary $A = (A_1, A_2)$, we say the $\mathsf{ind\text{-}so\text{-}enc}$-advantage of A with respect to $\mathcal{M}$ is

$$\mathbf{Adv}^{\mathsf{ind\text{-}so\text{-}enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) = 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda)] - 1,$$

where the $\mathsf{ind\text{-}so\text{-}enc}$ security experiment is defined in Figure 1, and $\mathcal{M}|_{I,\mathbf{m}_0[I]}$ returns a random n-vector $\mathbf{m}_1$ according to $\mathcal{M}$, subject to $\mathbf{m}_1[I] = \mathbf{m}_0[I]$. In other words, $\mathcal{M}|_{I,\mathbf{m}_0[I]}$ denotes conditionally resampling from the message space subject to the constraint that the messages corresponding to indices in I are equal to $\mathbf{m}_0[I]$.

We say that a public-key encryption scheme $\mathcal{AE}$ is $\mathsf{ind\text{-}so\text{-}enc}$-secure if for any efficient message sampler $\mathcal{M}$ that supports efficient conditional resampling and for all efficient adversaries A, the $\mathsf{ind\text{-}so\text{-}enc}$-advantage of A with respect to $\mathcal{M}$ is negligible in the security parameter.

In words, the experiment proceeds as follows. The adversary is given a public key pk and n ciphertexts $\mathbf{c}$ encrypted under public key pk. The messages corresponding to the n ciphertexts come from the joint distribution $\mathcal{M}$. The adversary then specifies a set I of t ciphertexts and receives the randomness $\mathbf{r}[I]$ used to generate those ciphertexts in addition to a message vector $\mathbf{m}_b$ such that $\mathbf{m}_b[I]$ were the actual messages encrypted using $\mathbf{r}[I]$ and the rest of $\mathbf{m}_b$ depends

$\underline{\textbf{Experiment } \mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda)}$
$\mathbf{m}_0 \leftarrow\!\!\$\ \mathcal{M}(1^\lambda);\ b \leftarrow\!\!\$\ \{0,1\};\ (pk, sk) \leftarrow\!\!\$\ \mathcal{K}(1^\lambda)$
For $i = 1, \ldots, n(\lambda)$ do
  $\mathbf{r}[i] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_0[i])$
  $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_0[i]; \mathbf{r}[i])$
$(I, \mathsf{st}) \leftarrow\!\!\$\ A_1(1^\lambda, pk, \mathbf{c})$
$\mathbf{m}_1 \leftarrow\!\!\$\ \mathcal{M}|_{I,\mathbf{m}_0[I]}$
$b' \leftarrow\!\!\$\ A_2(\mathsf{st}, \mathbf{r}[I], \mathbf{m}_b)$
Return $(b = b')$

Fig. 1. The IND-SO-ENC security experiment

on the bit b. If b, which the experiment chooses randomly, is 0, the rest of the messages in the vector are the actual messages used to create the ciphertexts $\mathbf{c}$ that were given to the adversary. If $b = 1$, the rest of the messages are instead resampled from $\mathcal{M}$, conditioned on I and $\mathbf{m}_b[I]$. The adversary must then try to guess the bit b.

The definition is a natural extension of ind-cpa to the selective decryption setting. Intuitively, the definition means that an adversary, after adaptively choosing to open some ciphertexts, cannot distinguish between the actual unopened messages and another set of messages that are equally likely given the opened messages that the adversary has seen.

Simulation-based. For any public-key encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, any message sampler $\mathcal{M}$, any relation R, any adversary $A = (A_1, A_2)$, and any simulator $S = (S_1, S_2)$, we say the sem-so-enc-advantage of A with respect to $\mathcal{M}$, R, and S is

$$\begin{aligned}\mathbf{Adv}^{\mathsf{ind\text{-}so\text{-}enc}}_{A,S,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) = &\Pr[\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}enc\text{-}real}}_{A,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) = 1] \\ &- \Pr[\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}enc\text{-}ideal}}_{S,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) = 1]\end{aligned}$$

where the sem-so-enc security experiments are defined in Figure 2.

We say that a public-key encryption scheme $\mathcal{AE}$ is sem-so-enc-secure if for any efficient message sampler $\mathcal{M}$, any efficiently computable relation R, and any efficient adversary A, there exists an efficient simulator S such that the sem-so-enc-advantage of A with respect to $\mathcal{M}$, R, and S is negligible in the security parameter.

$\underline{\textbf{Experiment } \mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}enc\text{-}real}}_{A,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda)}$	$\underline{\textbf{Experiment } \mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}enc\text{-}ideal}}_{S,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda)}$
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}(1^\lambda);\ (pk, sk) \leftarrow\!\!\$\ \mathcal{K}(1^\lambda)$	$\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$
For $i = 1, \ldots, n(\lambda)$ do	$(I, \mathsf{st}) \leftarrow\!\!\$\ S_1(1^\lambda)$
$\mathbf{r}[i] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}[i])$	$w \leftarrow\!\!\$\ S_2(\mathsf{st}, \mathbf{m}[I])$
$\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$	Return $\mathsf{R}(\mathbf{m}, w)$
$(I, \mathsf{st}) \leftarrow\!\!\$\ A_1(1^\lambda, pk, \mathbf{c})$	
$w \leftarrow\!\!\$\ A_2(\mathsf{st}, \mathbf{r}[I], \mathbf{m}[I])$	
Return $\mathsf{R}(\mathbf{m}, w)$	

Fig. 2. The two security experiments for SEM-SO-ENC

In words, the experiments proceed as follows. In the sem-so-enc-real experiment, the adversary A is given a public key pk and n ciphertexts $\mathbf{c}$ encrypted under public key pk. The messages corresponding to the n ciphertexts come from the joint distribution $\mathcal{M}$. The adversary then specifies a set I of t ciphertexts and receives the messages $\mathbf{m}[I]$ and randomness $\mathbf{r}[I]$ used to generate those ciphertexts. The adversary then outputs a string w and the output of the experiment is $\mathsf{R}(\mathbf{m}, w)$, the relation applied to the message vector and adversary's output. In the sem-so-enc-ideal experiment, a vector $\mathbf{m}$ of messages is chosen and the simulator, given only the security parameter, chooses a set I. The simulator is then given $\mathbf{m}[I]$, the messages corresponding to the index set I. Finally, the simulator outputs a string w and the output of the experiment is $\mathsf{R}(\mathbf{m}, w)$.

4 Lossy Encryption

The main tool we use in our results is what we call a *Lossy Encryption Scheme*. Informally, a lossy encryption scheme is a public-key encryption scheme with a standard key generation algorithm (which produces 'real' keys) and a lossy key generation algorithm (which produces 'lossy' keys), such that encryptions with real keys are committing, while encryptions with lossy keys are not committing. Peikert, Vaikuntanathan, and Waters [41] called such lossy keys "messy keys", for *me*ssage lo*ssy*, while defining a related notion called Dual-Mode Encryption. The notion of Lossy Encryption is also similar to Meaningful/Meaningless Encryption [34], formalized by Kol and Naor.

More formally, a *lossy public-key encryption scheme* $\mathcal{AE} = (\mathcal{K}, \mathcal{K}_{\mathsf{loss}}, \mathcal{E}, \mathcal{D})$ is a tuple of PT algorithms defined as follows. The key generation algorithm $\mathcal{K}$ takes as input the security parameter 1^λ and outputs a keypair (pk, sk); we call public keys generated by $\mathcal{K}$ real public keys. The lossy key generation algorithm $\mathcal{K}_{\mathsf{loss}}$ takes as input the security parameter and outputs a keypair (pk, sk); we call such pk lossy public keys. The encryption algorithm $\mathcal{E}$ takes as input a public key pk (either from $\mathcal{K}$ or $\mathcal{K}_{\mathsf{loss}}$) and a message m and outputs a ciphertext c. The decryption algorithm takes as input a secret key sk and a ciphertext c and outputs either a message m, or $\perp$ in the case of failure. We require the following properties from $\mathcal{AE}$:

1. *Correctness on real keys.* For all $(pk, sk) \leftarrow\!\!\$\, \mathcal{K}$ it must be the case that $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$. In other words, when the real key generation algorithm is used, the standard public-key encryption correctness condition must hold.
2. *Indistinguishability of real keys from lossy keys.* No polynomial-time adversary can distinguish between the first outputs of $\mathcal{K}$ and $\mathcal{K}_{\mathsf{loss}}$. We call the advantage of an adversary A distinguishing between the two the lossy-key-advantage of A and take it to mean the obvious thing, i.e., the probability that A outputs 1 when given the first output of $\mathcal{K}$ is about the same as the probability it outputs 1 when given the first output of $\mathcal{K}_{\mathsf{loss}}$.
3. *Lossiness of encryption with lossy keys.* For any $(pk, sk) \leftarrow \mathcal{K}_{\mathsf{loss}}$ and two distinct messages m_0, m_1, it must be the case that $\mathcal{E}(pk, m_0) \equiv_s \mathcal{E}(pk, m_1)$. We say the advantage of an adversary A in distinguishing between the two

is the lossy-ind advantage of A and take it to mean the advantage of A in the standard ind-cpa game *when the public key pk in the ind-cpa game is lossy.* Notice that because the ciphertexts are *statistically* close, even an unbounded distinguisher will have low advantage. We sometimes call ciphertexts created with lossy public keys *lossy ciphertexts.*

4. *Possible to claim any plaintext.* There exists a (possibly unbounded) algorithm Opener that, given a lossy public key pk_{loss}, message m, and ciphertext $c = \mathcal{E}(pk_{\mathsf{loss}}, m)$, will output $r' \in_R \mathsf{Coins}_{\mathcal{E}}(pk_{\mathsf{loss}}, m)$ such that $\mathcal{E}(pk_{\mathsf{loss}}, m; r') = c$. In other words, Opener will find correctly distributed randomness to open a lossy ciphertext to the plaintext it encrypts. It then directly follows from the lossiness of encryption that with high probability the opener algorithm can successfully open *any* ciphertext to *any* plaintext.

We note that the fourth property is already implied by the first three properties; the canonical (inefficient) Opener algorithm will, given pk_{loss}, m, and c, simply try all possible coins to find the set of all r such that $\mathcal{E}(pk_{\mathsf{loss}}, m; r) = c$ and output a random element of that set. Nevertheless, we explicitly include the property because it is convenient in the proofs, and later we will consider variations of the definition which consider other (more efficient) opener algorithms.

We also note that the definition of lossy encryption already implies ind-cpa security. We next provide two instantiations of lossy public-key encryption, one from DDH and one from lossy trapdoor functions.

4.1 Instantiation from DDH

We now describe a lossy public-key encryption scheme based on the DDH assumption. Recall that the DDH assumption for cyclic group $\mathbb{G}$ of order prime p says that for random generator $g \in \mathbb{G}^*$ (we use $\mathbb{G}^*$ to denote the generators of $\mathbb{G}$), the tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) are computationally indistinguishable, where $a, b, c \leftarrow\!\!\$\ \mathbb{Z}_p$.

The scheme we describe below is originally from [36], yet some of our notation is taken from the similar dual-mode encryption scheme of [41]. The scheme has structure similar to ElGamal.

Let $\mathbb{G}$ be a prime order group of order prime p. The scheme $\mathcal{AE}_{\mathsf{ddh}} = (\mathcal{K}, \mathcal{K}_{\mathsf{loss}}, \mathcal{E}, \mathcal{D})$ is a tuple of polynomial-time algorithms defined as follows:

Algorithm $\mathcal{K}(1^\lambda)$
$g \leftarrow\!\!\$\ \mathbb{G}^*;\ x, r \leftarrow\!\!\$\ \mathbb{Z}_p$
$pk \leftarrow (g, g^r, g^x, g^{rx})$
$sk \leftarrow x$
Return (pk, sk)

Algorithm $\mathcal{E}(pk, m)$
$(g, h, g', h') \leftarrow pk$
$(u, v) \leftarrow\!\!\$\ \mathsf{Rand}(g, h, g', h')$
Return $(u, v \cdot m)$

Algorithm $\mathcal{D}(sk, c)$
$(c_0, c_1) \leftarrow c$
Return c_1 / c_0^{sk}

Algorithm $\mathcal{K}_{\mathsf{loss}}(1^\lambda)$
$g \leftarrow\!\!\$\ \mathbb{G}^*;\ r, x \neq y \leftarrow\!\!\$\ \mathbb{Z}_p$
$pk \leftarrow (g, g^r, g^x, g^{ry})$
$sk \leftarrow \perp$
Return (pk, sk)

Subroutine $\mathsf{Rand}(g, h, g', h')$
$s, t \leftarrow\!\!\$\ \mathbb{Z}_p$
$u \leftarrow g^s h^t;\ v \leftarrow (g')^s (h')^t$
Return (u, v)

We show that $\mathcal{AE}_{\mathsf{ddh}}$ satisfies the four properties of lossy encryption schemes.

1. *Correctness on real keys.* To see the correctness property is satisfied, consider a (real) public key $pk = (g, g^r, g^x, g^{rx})$ and corresponding secret key $sk = x$. Then, for some message $m \in \mathbb{G}$

$$\begin{aligned}\mathcal{D}(sk, \mathcal{E}(pk, m)) &= \mathcal{D}(sk, (g^{s+rt}, g^{xs+rxt} \cdot m)) \\ &= (g^{xs+rxt} \cdot m)/(g^{s+rt})^x \\ &= m\end{aligned}$$

2. *Indistinguishability of real keys from lossy keys.* This follows from the assumption that DDH is hard in the groups we are using, since the first output of $\mathcal{K}$ is (g, g^r, g^x, g^{rx}) and the first output of $\mathcal{K}_{\mathsf{loss}}$ is (g, g^r, g^x, g^{ry}) for $y \neq x$.
3. *Lossiness of encryption with lossy keys.* We need to show that for any lossy public key pk generated by $\mathcal{K}_{\mathsf{loss}}$, and any messages $m_0 \neq m_1 \in \mathbb{G}$, it is the case that $\mathcal{E}(pk, m_0) \equiv_s \mathcal{E}(pk, m_1)$. The results of Peikert, Vaikuntanathan, and Waters can be applied here (specifically Lemma 4 from their paper [41]). We repeat their lemma for completeness.

   Lemma 1 (Lemma 4 from [41]). *Let $\mathbb{G}$ be an arbitrary multiplicative group of prime order p. For each $x \in \mathbb{Z}_p$, define $\mathsf{DLOG}_{\mathbb{G}}(x) = \{(g, g^x) : g \in \mathbb{G}\}$. There is a probabilistic algorithm Rand that takes generators $g, h \in \mathbb{G}$ and elements $g', h' \in \mathbb{G}$, and outputs a pair $(u, v) \in \mathbb{G}^2$ such that:*
   - *If $(g, g'), (h, h') \in \mathsf{DLOG}_{\mathbb{G}}(x)$ for some x, then (u, v) is uniformly random in $\mathsf{DLOG}_{\mathbb{G}}(x)$.*
   - *If $(g, g') \in \mathsf{DLOG}_{\mathbb{G}}(x)$ and $(h, h') \in \mathsf{DLOG}_{\mathbb{G}}(y)$ for $x \neq y$, then (u, v) is uniformly random in $\mathbb{G}^2$.*

   The Rand procedure mentioned in the lemma is exactly our Rand procedure defined above. As [41] proves, this lemma shows that encryptions under a lossy key are statistically close, since such encryptions are just pairs of uniformly random group elements.
4. *Possible to claim any plaintext.* The unbounded algorithm Opener is simply the canonical opener mentioned above. Specifically, on input lossy public key $pk = (g, h, g', h')$, message $m \in \mathbb{G}$, and ciphertext $(c_1, c_2) \in \mathbb{G}^2$, it computes the set of all $s, t \in \mathbb{Z}_p$ such that $\mathsf{Rand}(g, h, g', h'; s, t)$ outputs $(c_1, c_2/m)$. It then outputs a random element of this set.

4.2 Instantiation from Lossy TDFs

Before giving our scheme we will recall a few definitions.

Definition 1 (Pairwise Independent Function Family). *A family of functions $\mathcal{H}_{n,m}$ from $\{0,1\}^n$ to $\{0,1\}^m$ is* pairwise-independent *if for any distinct $x, x' \in \{0,1\}^n$ and any $y, y \in \{0,1\}^m$,*

$$\Pr_{h \xleftarrow{\$} \mathcal{H}_{n,m}}[h(x) = y \wedge h(x') = y'] = \frac{1}{2^{2m}}.$$

For our results, we make use of *lossy trapdoor functions*, a primitive recently introduced by Peikert and Waters [40]. Informally, a lossy trapdoor function is similar to a traditional injective trapdoor function, but with the extra property that the trapdoor function is indistinguishable from another function that loses information about its input. We recall the definition from Peikert and Waters (with minor notational changes):

Definition 2 (Collection of (n,k) Lossy Trapdoor Functions). *Let λ be a security parameter, $n = n(\lambda) = \mathsf{poly}(\lambda)$, and $k = k(\lambda) \leq n$. A* collection of (n,k)-lossy trapdoor functions $\mathcal{L}_{n,k} = (S_{\mathsf{tdf}}, S_{\mathsf{loss}}, F_{\mathsf{tdf}}, F_{\mathsf{tdf}}^{-1})$ *is a tuple of algorithms with the following properties:*

1. Easy to sample, compute, and invert given a trapdoor, an injective trapdoor function. *The sampler S_{tdf}, on input 1^λ outputs (s,t), algorithm F_{tdf}, on input index s and some point $x \in \{0,1\}^n$, outputs $f_s(x)$, and algorithm F_{tdf}^{-1}, on input t and y outputs $f_s^{-1}(y)$.*
2. Easy to sample and compute lossy functions. *Algorithm S_{loss}, on input 1^λ, outputs $(s,\perp)$, and algorithm F_{tdf}, on input index s and some point $x \in \{0,1\}^n$, outputs $f_s(x)$, and the image size of f_s is at most $2^r = 2^{n-k}$.*
3. Difficult to distinguish between injective and lossy. *The function indices outputted by the sampling algorithms S_{tdf} and S_{loss} should be computationally indistinguishable. We say the advantage of distinguishing between the indices is the* ltdf*-advantage.*

We now describe an instantiation of lossy encryption based on lossy trapdoor functions.

Let λ be a security parameter and let $(S_{\mathsf{tdf}}, S_{\mathsf{loss}}, F_{\mathsf{tdf}}, F_{\mathsf{tdf}}^{-1})$ define a collection of (n,k)-lossy trapdoor functions. Also let $\mathcal{H}$ be a collection of pairwise independent hash functions from n bits to ℓ bits; the message space of the cryptosystem will then be $\{0,1\}^\ell$. The parameter ℓ should be such that $\ell \leq k - 2\log(1/\delta)$, where δ is a negligible function in the security parameter λ. The scheme $\mathcal{AE}_{\mathsf{loss}} = (\mathcal{K}, \mathcal{K}_{\mathsf{loss}}, \mathcal{E}, \mathcal{D})$ is then defined as follows:

Algorithm $\mathcal{K}(1^\lambda)$	**Algorithm** $\mathcal{E}(pk, m)$	**Algorithm** $\mathcal{D}(sk, c)$
$(s,t) \leftarrow\!\!\$\ S_{\mathsf{tdf}}(1^\lambda)$	$(s,h) \leftarrow pk$	$(t,h) \leftarrow sk$
$h \leftarrow\!\!\$\ \mathcal{H}$	$x \leftarrow\!\!\$\ \{0,1\}^n$	$(c_1,c_2) \leftarrow c$
$pk \leftarrow (s,h)$; $sk \leftarrow (t,h)$	$c_1 \leftarrow F_{\mathsf{tdf}}(s,x)$	$x \leftarrow F_{\mathsf{tdf}}^{-1}(t,c_1)$
Return (pk, sk)	$c_2 \leftarrow m \oplus h(x)$	Return $h(x) \oplus c_2$
	Return (c_1, c_2)	

The $\mathcal{K}_{\mathsf{loss}}$ algorithm is simply the same as $\mathcal{K}$, but using S_{loss} instead of S_{tdf}. (In this case, the trapdoor t will be $\perp$.)

We now show that $\mathcal{AE}_{\mathsf{loss}}$ satisfies the four properties of lossy encryption schemes.

1. *Correctness on real keys.* This follows since when $pk = (s,h)$ was generated by $\mathcal{K}$, s is such that $(s,t) \leftarrow\!\!\$\ S_{\mathsf{tdf}}(1^\lambda)$ and $h \leftarrow\!\!\$\ \mathcal{H}$ so that

$$\begin{aligned}\mathcal{D}(sk, \mathcal{E}(pk, m)) &= h(F_{\mathsf{tdf}}^{-1}(t, F_{\mathsf{tdf}}(s,x))) \oplus (m \oplus h(x)) \\ &= h(x) \oplus m \oplus h(x) \\ &= m\end{aligned}$$

2. *Indistinguishability of real keys from lossy keys.* We need to show that any efficient adversary has low $\mathsf{lossy\text{-}key}$ advantage in distinguishing between a real public key (s, h) and a lossy key (s', h'), where $(s, h) \leftarrow\!\!\$\ \mathcal{K}(1^\lambda)$ and $(s', h') \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{loss}}(1^\lambda)$. Since s is the first output of S_{tdf} and s' is the first output of S_{loss}, we use the third property of lossy trapdoor functions, specifically that the function indices outputted by S_{tdf} and S_{loss} are computationally indistinguishable.
3. *Lossiness of encryption with lossy keys.* We need to show that for any lossy public key pk generated by $\mathcal{K}_{\mathsf{loss}}$, and any messages $m_0 \neq m_1 \in \{0,1\}^\ell$, it is the case that $\mathcal{E}(pk, m_0) \equiv_s \mathcal{E}(pk, m_1)$. As Peikert and Waters show in [40], this is true because of the lossiness of f_s (where s is part of pk, generated by S_{loss}). Specifically, they show that the average min-entropy $\tilde{\mathrm{H}}_\infty(x|(c_1, pk))$ of the random variable x, given $f_s(x)$ and pk is at least k, and since $\ell \leq k - 2\log(1/\delta)$, it follows that $h(x)$ will be δ-close to uniform and $m_b \oplus h(x)$ will also be δ-close to uniform for either bit b.
4. *Possible to claim any plaintext.* Again, the opener is simply the canonical opener that is guaranteed to be correct by the first three properties. Specifically, the (unbounded) algorithm Opener, on input a public key $pk = (s, h)$, message $m' \in \{0,1\}^\ell$, and ciphertext $c = (c_1, c_2) = (f_s(x), h(x) \oplus m)$ for some $x \in \{0,1\}^n$ and $m \in \{0,1\}^\ell$, must output $x' \in \{0,1\}^n$ such that $f_s(x') = c_1$ and $h(x') \oplus m' = c_2$. To do so, Opener enumerates over all $\{0,1\}^n$ and creates a set $X = \{x' \in \{0,1\}^n \ : \ f_s(x') = c_1 \wedge h(x') = m' \oplus c_2\}$ before returning a random $x \in X$.

4.3 An Extension: Efficient Opening

Recall that in the above definition of lossy encryption, the Opener algorithm could be unbounded. We will now consider a refinement of the definition that will be useful for achieving the simulation-based selective opening definition. We say that a PKE scheme $\mathcal{AE}$ is a *lossy encryption scheme with efficient opening* if it satisfies the following four properties:

1. *Correctness on real keys.* For all $(pk, sk) \leftarrow\!\!\$\ \mathcal{K}$ it must be the case that $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$.
2. *Indistinguishability of real keys from lossy keys.* No polynomial-time adversary can distinguish between the first outputs of $\mathcal{K}$ and $\mathcal{K}_{\mathsf{loss}}$.
3. *Lossiness of encryption with lossy keys.* For any $(pk, sk) \leftarrow \mathcal{K}_{\mathsf{loss}}$ and two distinct messages m_0, m_1, it must be the case that $\mathcal{E}(pk, m_0) \equiv_i \mathcal{E}(pk, m_1)$. Notice that we require ciphertexts to be identically distributed.
4. *Possible to efficiently claim any plaintext.* There exists an efficient algorithm Opener that on input lossy keys sk_{loss} and pk_{loss}, message m', and ciphertext $c = \mathcal{E}(pk_{\mathsf{loss}}, m)$, outputs an $r' \in_R \mathsf{Coins}_{\mathcal{E}}(pk_{\mathsf{loss}}, m')$ such that $\mathcal{E}(pk_{\mathsf{loss}}, m'; r') = c$. In words, the algorithm Opener is able to open ciphertexts to arbitrary plaintexts efficiently.

We emphasize that it is important for the opener algorithm to take as input the lossy secret key. This may seem strange, since in the two schemes described above the lossy secret key was simply $\perp$, but this need not be the case.

4.4 The GM Probabilistic Encryption Scheme

The Goldwasser-Micali Probabilistic encryption scheme [27] is an example of a lossy encryption scheme with efficient opening. We briefly recall the GM scheme. Let Par be an algorithm that efficiently chooses two large random primes p and q and outputs them along with their product N. Let $\mathcal{J}_p(x)$ denote the Jacobi symbol of x modulo p. We denote by QR_N the group of quadratic residues modulo N and we denote by QNR_N^{+1} the group of quadratic non-residues x such that $\mathcal{J}_N(x) = +1$. Recall that the security of the GM scheme is based on the Quadratic Residuosity Assumption, which states that it is difficult to distinguish a random element of QR_N from a random element of QNR_N^{+1}. The scheme $\mathcal{AE}_{GM} = (\mathcal{K}, \mathcal{K}_{\mathsf{loss}}, \mathcal{E}, \mathcal{D})$ is defined as follows.

Algorithm $\mathcal{K}(1^\lambda)$	**Algorithm** $\mathcal{E}(pk, m)$	**Algorithm** $\mathcal{D}(sk, \mathbf{c})$
$(N, p, q) \leftarrow\!\!\$\ \mathsf{Par}(1^\lambda)$	$(N, x) \leftarrow pk$	$(p, q) \leftarrow sk$
$x \leftarrow\!\!\$\ \mathsf{QNR}_N^{+1}$	For $i = 1$ to $\lvert m\rvert$	For $i = 1$ to $\lvert\mathbf{c}\rvert$
$pk \leftarrow (N, x)$	$r_i \leftarrow\!\!\$\ \mathbb{Z}_N^*$	If $\mathcal{J}_p(\mathbf{c}[i]) = \mathcal{J}_q(\mathbf{c}[i]) = +1$
$sk \leftarrow (p, q)$	$\mathbf{c}[i] \leftarrow r_i^2 \cdot x^{m_i} \bmod N$	$m_i \leftarrow 0$
Return (pk, sk)	Return $\mathbf{c}$	Else $m_i \leftarrow 1$
		Return m

The algorithm $\mathcal{K}_{\mathsf{loss}}$ is the same as $\mathcal{K}$ except that x is chosen at random from QR_N instead of QNR_N^{+1}; in the lossy case the secret key is still the factorization of N.

It is easy to see that the scheme $\mathcal{AE}_{GM}$ meets the first three properties of lossy PKE schemes with efficient opening: the correctness of the scheme under real keys was shown in [27], the indistinguishability of real keys from lossy keys follows directly from the Quadratic Residuosity Assumption, and encryptions under lossy keys are lossy since in that case all ciphertexts are just sequences of random quadratic residues. We claim that $\mathcal{AE}_{GM}$ is also efficiently openable. To see this consider the (efficient) algorithm Opener that takes as input secret key $sk = (p, q)$, public key $pk = (N, x)$, plaintext m, and encryption $\mathbf{c}$. For simplicity, say m has length n bits. For each $i \in [n]$, Opener uses p and q to efficiently compute the four square roots of $\mathbf{c}[i]/x^{m_i}$ and lets $\mathbf{r}[i]$ be a randomly chosen one of the four. The output of Opener is the sequence $\mathbf{r}$, which is just a sequence of random elements in $\mathbb{Z}_N^*$.

5 SOA-Security from Lossy Encryption

We now state our main results for encryption: any lossy public-key encryption scheme is ind-so-enc-secure, and any lossy public-key encryption scheme with efficient opening is sem-so-enc-secure.

Theorem 1 (Lossy Encryption implies IND-SO-ENC security). *Let λ be a security parameter, $\mathcal{AE} = (\mathcal{K}, \mathcal{K}_{lossy}, \mathcal{E}, \mathcal{D})$ be any* lossy public-key encryption scheme*, $\mathcal{M}$ any efficiently samplable distribution that supports efficient re-sampling, and A be any polynomial-time adversary corrupting $t = t(\lambda)$ parties.*

Then, there exists an unbounded lossy-ind *adversary* C *and an efficient* lossy-key *adversary* B *such that*

$$\mathbf{Adv}^{\textsf{ind-so-enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) \leq 2n \cdot \mathbf{Adv}^{\textsf{lossy-ind}}_{C,\mathcal{AE}}(\lambda) + 2 \cdot \mathbf{Adv}^{\textsf{lossy-key}}_{B,\mathcal{AE}}(\lambda).$$

Proof. We will prove the theorem using a sequence of game transitions. We start with a game that is simply the ind-so-enc experiment run with A, and end with a game in which A has no advantage, showing that each subsequent game is either computationally or statistically indistinguishable from the previous game. Now, we know that

$$\mathbf{Adv}^{\textsf{ind-so-enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) = 2\Pr[\mathbf{Exp}^{\textsf{ind-so-enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda)] - 1$$

by the definition of ind-so-enc-security (see Section 3.2). We will now explain the game transitions.

- G_0: The same as the ind-so-enc experiment.
- G_1: The only change is that the A_1 is given a lossy public key and lossy ciphertexts.
- H_0: Instead of opening the ciphertexts corresponding to index I (provided by A_1) by revealing the actual coins used to generate the ciphertexts, H_0 runs the Opener algorithm on the actual messages and ciphertexts and gives A_2 the coins outputted. By the definition of the Opener algorithm (see Section 4), the coins will be correctly distributed and consistent with the ciphertexts.
- H_j: We generalize H_0 with a sequence of hybrid games. In the jth hybrid game, the first j ciphertexts given to A_1 are encryptions of dummy messages instead of the first j messages outputted by $\mathcal{M}$. Yet, the game still opens the ciphertexts for A_2 *to the actual messages produced by* $\mathcal{M}$ using the Opener algorithm.
- H_n: In the last hybrid game, A_1 is given encryptions of only the dummy message, yet A_2 receives openings of the ciphertexts to the actual messages generated by $\mathcal{M}$.

We first claim that there is an efficient adversary B such that

$$\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1] = \mathbf{Adv}^{\textsf{lossy-key}}_{B,\mathcal{AE}}(\lambda). \tag{1}$$

To see this consider a B that is given a challenge public key pk^* and must decide whether or not it is lossy. The adversary uses the ind-so-enc-adversary A and executes exactly the same as G_0 and G_1, giving the adversary the challenge key pk^* and ciphertexts generated using pk^*. It is important for the conditional resamplability of $\mathcal{M}$ to be efficient in order for adversary B to be efficient.

Next, we claim that

$$\Pr[\mathsf{G}_1] = \Pr[\mathsf{H}_0]. \tag{2}$$

Recall that H_0 opens ciphertexts $\mathbf{c}[i] = \mathcal{E}(pk, \mathbf{m}_0[i])$ by using the Opener procedure. The key point is that in H_0, $\mathbf{c}[i]$ is still opened to $\mathbf{m}_0[i]$. This ensures us that Opener will always succeed in finding coins that open the ciphertext

correctly, and ensures us that the output of Opener is identically distributed to the actual coins used to encrypt $\mathbf{m}$. Thus, the claim follows.

We can now use a standard hybrid arguments to claim there is an *unbounded* adversary C such that

$$\Pr[\mathsf{H}_0] - \Pr[\mathsf{H}_n] = n \cdot \mathbf{Adv}^{\mathsf{lossy\text{-}ind}}_{C,\mathcal{AE}}(\lambda). \tag{3}$$

Adversary C, on input a lossy public key pk^*, will operate the same as H_j (for some guess j) except that it will use the challenge key, and for the jth ciphertext it will use the result of issuing an IND-CPA challenge consisting of the dummy message $\mathbf{m}_{dum}$ and the real message $\mathbf{m}_0[j]$. The adversary C needs to be unbounded because it runs the (possibly inefficient) procedure Opener. With standard IND-CPA, the unbounded nature of C would be problematic. However, in the case of lossy encryption, the encryptions of two distinct lossy ciphertexts are *statistically close* instead of just computationally indistinguishable, so C will still have only negligible advantage.

Finally, we claim that

$$\Pr[\mathsf{H}_n] = 1/2, \tag{4}$$

which is true since in H_n the adversary A_1 is given encryptions of dummy messages and has no information about the messages chosen from $\mathcal{M}$. (In fact, we could modify the games again and move the choice of the messages to after receiving I from A_1.)

Combining the above equations, we see that

$$\mathbf{Adv}^{\mathsf{ind\text{-}sda}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) \leq 2n \cdot \mathbf{Adv}^{\mathsf{lossy\text{-}ind}}_{C,\mathcal{AE}}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{lossy\text{-}key}}_{B,\mathcal{AE}}(\lambda),$$

which proves the theorem. □

Theorem 2 (Lossy Encryption with Efficient Opening implies SEM-SO-ENC security). *Let λ be a security parameter, $\mathcal{AE} = (\mathcal{K}, \mathcal{K}_{lossy}, \mathcal{E}, \mathcal{D})$ be any* lossy public-key encryption scheme with efficient opening*, $\mathcal{M}$ any efficiently samplable distribution, R an efficiently computable relation, and $A = (A_1, A_2)$ be any polynomial-time adversary corrupting $t = t(\lambda)$ parties. Then, there exists an efficient simulator $S = (S_1, S_2)$ and efficient* lossy-key *adversary B such that*

$$\mathbf{Adv}^{\mathit{sem\text{-}so\text{-}enc}}_{A,S,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \leq \mathbf{Adv}^{\mathit{lossy\text{-}key}}_{B,\mathcal{AE}}(\lambda).$$

Proof (Sketch). The proof of Theorem 2 is very similar to the proof of Theorem 1, so we will only sketch it here. For more details see [7]. We can modify the sem-so-enc-real experiment step by step until we have a successful simulator in the sem-so-enc-ideal experiment. Consider the following sequence of games:

G_0: The sem-so-enc-real experiment.
G_1: Same as G_0 except the adversary A_1 is given a lossy public key. The games are indistinguishable by the second property of efficiently openable lossy encryption.
G_2: Instead of giving A_2 the actual randomness $\mathbf{r}[I]$, the experiment uses the efficient Opener procedure.
G_3: Adversary A_1 is given encryptions of dummy messages, but A_2 is still given openings to the actual messages in $\mathbf{m}$. To do this, the efficient Opener algorithm is applied to the dummy ciphertexts.

We can then construct a simulator $S = (S_1, S_2)$ that runs A exactly as its run in G_3. Specifically, S chooses a lossy keypair and runs A_1 with a vector of encryptions of dummy messages. When A_1 outputs a set I, S asks for the same set I and learns messages $\mathbf{m}_I$. The simulator then uses the efficient Opener algorithm to open the dummy ciphertexts to the values $\mathbf{m}_I$ and finally outputs the same w as A_2. Thus, the game G_3 is identical to the sem-so-enc-ideal experiment run with simulator S. Since all of the games are close, the theorem follows. □

6 Commitment Preliminaries and Definitions

Commitment schemes

Definition 3 (Commitment scheme). *For a pair of PPT machines* $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ *and a machine* A*, consider the following experiments:*

Experiment $\mathbf{Exp}^{\mathsf{binding}}_{\mathsf{Com},A}(\lambda)$	**Experiment** $\mathbf{Exp}^{\mathsf{hiding}\text{-}b}_{\mathsf{Com},A}(\lambda)$
run $\langle \mathsf{R}(\mathtt{recv}), A(\mathtt{com})\rangle$	$(m_0, m_1) \leftarrow\!\!\$\ A(\mathtt{choose})$
$m'_0 \leftarrow\!\!\$\ \langle \mathsf{R}(\mathtt{open}), A(\mathtt{open}, 0)\rangle$	return $\langle A(\mathtt{recv}), \mathsf{S}(\mathtt{com}, m_b)\rangle$
rewind A and R back to after step 1	
$m'_1 \leftarrow\!\!\$\ \langle \mathsf{R}(\mathtt{open}), A(\mathtt{open}, 1)\rangle$	
return 1 iff $\bot \neq m'_0 \neq m'_1 \neq \bot$	

In this, $\langle A, \mathsf{S}\rangle$ *denotes the output of* A *after interacting with* S*, and* $\langle \mathsf{R}, A\rangle$ *denotes the output of* R *after interacting with* A*. We say that* Com *is a* commitment scheme *iff the following holds:*

Syntax. *For any* $m \in \{0,1\}^\lambda$*,* $\mathsf{S}(\mathtt{com}, m)$ *first interacts with* $\mathsf{R}(\mathtt{recv})$*. We call this the* commit phase*. After that,* $\mathsf{S}(\mathtt{open})$ *interacts again with* $\mathsf{R}(\mathtt{open})$*, and* R *finally outputs a value* $m' \in \{0,1\}^\lambda \cup \{\bot\}$*. We call this the* opening phase*.*

Correctness. *We have* $m' = m$ *always and for all* m*.*

Hiding. *For a PPT machine* A*, let*

$$\mathbf{Adv}^{\mathsf{hiding}}_{\mathsf{Com},A}(\lambda) := \Pr\left[\mathbf{Exp}^{\mathsf{hiding}\text{-}0}_{\mathsf{Com},A} = 1\right](\lambda) - \Pr\left[\mathbf{Exp}^{\mathsf{hiding}\text{-}1}_{\mathsf{Com},A} = 1\right](\lambda),$$

where $\mathbf{Exp}^{\mathsf{hiding}\text{-}b}_{\mathsf{Com},A}$ *is depicted below. For* Com *to be hiding, we demand that* $\mathbf{Adv}^{\mathsf{hiding}}_{\mathsf{Com},A}$ *is negligible for all PPT* A *that satisfy* $m_0, m_1 \in \{0,1\}^\lambda$ *always.*

Binding. *For a machine A, consider the experiment $\mathbf{Exp}^{\mathsf{binding}}_{\mathsf{Com},A}$ below. For* Com *to be binding, we require that $\mathbf{Adv}^{\mathsf{binding}}_{\mathsf{Com},A}(\lambda) = \Pr\left[\mathbf{Exp}^{\mathsf{binding}}_{\mathsf{Com},A}(\lambda) = 1\right]$ is negligible for all PPT A.*

Further, we say that Com *is* perfectly binding *iff $\mathbf{Adv}^{\mathsf{binding}}_{\mathsf{Com},A} = 0$ for all A. We say that* Com *is* statistically hiding *iff $\mathbf{Adv}^{\mathsf{hiding}}_{\mathsf{Com},A}$ is negligible for all (not necessarily PPT) A.*

Definition 4 (Non-interactive commitment scheme). *A* non-interactive commitment scheme *is a commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ in which both commit and opening phase consist of only one message sent from S to R. We can treat a non-interactive commitment scheme as a pair of algorithms rather than machines. Namely, we write $(com, dec) \leftarrow\!\!\$\ \mathsf{S}(m)$ shorthand for the commit message com and opening message dec sent by S on input m. We also denote by $m' \leftarrow\!\!\$\ \mathsf{R}(com, dec)$ the final output of R upon receiving com in the commit phase and dec in the opening phase.*

Note that perfectly binding implies that *any* commitment can only be opened to at most one value m. Perfectly binding (non-interactive) commitment schemes can be achieved from any one-way permutation (e.g., Blum [9]). On the other hand, statistically hiding implies that for any $m_0, m_1 \in \{0,1\}^\lambda$, the statistical distance between the respective views of the receiver in the commit phase is negligible. One-way functions suffice to implement statistically hiding (interactive) commitment schemes (Haitner and Reingold [29]), but there are certain lower bounds for the communication complexity of such constructions (Wee [47], Haitner et al.[30]). However, if we assume the existence of (families of) collision-resistant hash functions, then even constant-round statistically hiding commitment schemes exist (Damgard et al. [19], Naor and Yung [37]).

Interactive argument systems and zero-knowledge. We recall some basic definitions concerning interactive argument systems, mostly following Goldreich [25].

Definition 5 (Interactive proof/argument system). *An* interactive proof system *for a language $\mathcal{L}$ with witness relation $\mathcal{R}$ is a pair of PPT machines $\mathsf{IP} = (\mathsf{P}, \mathsf{V})$ such that the following holds:*

Completeness. *For every family $(x_\lambda, w_\lambda)_{\lambda\in\mathbb{N}}$ such that $\mathcal{R}(x_\lambda, w_\lambda)$ for all λ and $|x_\lambda|$ is polynomial in λ, we have that the probability for $\mathsf{V}(x_\lambda)$ to output 1 after interacting with $\mathsf{P}(x_\lambda, w_\lambda)$ is at least $2/3$.*

Soundness. *For every machine P^* and every family $(x_\lambda, z_\lambda)_{\lambda\in\mathbb{N}}$ such that $|x_\lambda| = \lambda$ and $x_\lambda \notin \mathcal{L}$ for all λ, we have that the probability for $\mathsf{V}(x_\lambda)$ to output 1 after interacting with $P^*(x_\lambda, z_\lambda)$ is at most $1/3$.*

If the soundness condition holds for all PPT machines P^* (but not necessarily for all unbounded P^*), then IP *is an* interactive argument system. *We say that* IP *enjoys* perfect completeness *if* V *always outputs 1 in the completeness condition.*

Furthermore, IP *has* negligible soundness error *if* V *outputs 1 only with negligible probability in the soundness condition.*

Definition 6 (Zero-knowledge). *Let* $\mathsf{IP} = (\mathsf{P}, \mathsf{V})$ *be an interactive proof or argument system for language* $\mathcal{L}$ *with witness relation* $\mathcal{R}$. IP *is* zero-knowledge *if for every PPT machine* V^**, there exists a PPT machine* S^* *such that for all sequences* $(x, w) = (x_\lambda, w_\lambda)_{\lambda\in\mathbb{N}}$ *with* $\mathcal{R}(x_\lambda, w_\lambda)$ *for all* λ *and* $|x_\lambda|$ *polynomial in* λ*, for all PPT machines* D*, and all auxiliary inputs* $z^{V^*} = (z_\lambda^{V^*})_{\lambda\in\mathbb{N}} \in (\{0,1\}^*)^{\mathbb{N}}$ *and* $z^D = (z_\lambda^D)_{\lambda\in\mathbb{N}} \in (\{0,1\}^*)^{\mathbb{N}}$*, we have that*

$$\mathsf{Adv}^{\mathsf{ZK}}_{V^*,S^*,(x,w),D,z^{V^*},z^D}(\lambda) := \Pr\Big[D(x_\lambda, z_\lambda^D, \langle \mathsf{P}(x_\lambda, w_\lambda), V^*(x_\lambda, z_\lambda^{V^*})\rangle) = 1\Big] - \Pr\Big[D(x_\lambda, z_\lambda^D, S^*(x_\lambda, z_\lambda^{V^*})) = 1\Big]$$

is negligible in λ*. Here* $\langle \mathsf{P}(x_\lambda, w_\lambda), V^*(x_\lambda, z_\lambda^{V^*})\rangle$ *denotes the transcript of the interaction between the prover* P *and* V^**.*

Most known interactive proof system achieve perfect completeness. Conversely, most systems do not enjoy a negligible soundness error "by nature"; their soundness has to be amplified via repetition, e.g., via sequential or concurrent composition. Thus, it is important to consider the concurrent composition of an interactive argument system:

Definition 7 (Concurrent zero-knowledge). *Let* $\mathsf{IP} = (\mathsf{P}, \mathsf{V})$ *be an interactive proof or argument system for language* $\mathcal{L}$ *with witness relation* $\mathcal{R}$. IP *is* zero-knowledge under concurrent composition *iff for every polynomial* $n = n(\lambda)$ *and PPT machine* V^**, there exists a PPT machine* S^* *such that for all sequences* $(x, w) = (x_{i,\lambda}, w_{i,\lambda})_{\lambda\in\mathbb{N}, i\in[n]}$ *with* $\mathcal{R}(x_{i,\lambda}, w_{i,\lambda})$ *for all* i, λ *and* $|x_{i,\lambda}|$ *polynomial in* λ*, for all PPT machines* D*, and all auxiliary inputs* $z^{V^*} = (z_\lambda^{V^*})_{\lambda\in\mathbb{N}} \in (\{0,1\}^*)^{\mathbb{N}}$ *and* $z^D = (z_\lambda^D)_{\lambda\in\mathbb{N}} \in (\{0,1\}^*)^{\mathbb{N}}$*, we have that*

$$\mathsf{Adv}^{\mathsf{cZK}}_{V^*,S^*,(x,w),D,z^{V^*},z^D} := \Pr\Big[D((x_{i,\lambda})_{i\in[n]}, z_\lambda^D, \langle \mathsf{P}((x_{i,\lambda}, w_{i,\lambda})_{i\in[n]}), V^*((x_{i,\lambda})_{i\in[n]}, z_\lambda^{V^*})\rangle) = 1\Big] - \Pr\Big[D((x_{i,\lambda})_{i\in[n]}, z_\lambda^D, S^*((x_{i,\lambda})_{i\in[n]}, z_\lambda^{V^*})) = 1\Big]$$

is negligible in λ*. Here* $\langle \mathsf{P}((x_{i,\lambda}, w_{i,\lambda})_{i\in[n]}), V^*((x_{i,\lambda})_{i\in[n]}, z_\lambda^{V^*})\rangle$ *denotes the transcript of the interaction between* n *copies of the prover* P *(with the respective inputs* $(x_{i,\lambda}, w_{i,\lambda})$ *for* $i = 1, \ldots, n$*) on the one hand, and* V^* *on the other hand.*

There exist interactive proof systems (with perfect completeness and negligible soundness error) that achieve Definition 7 for arbitrary NP-languages if one-way permutations exist (e.g., Richardson and Kilian [44]; see also [33, 15, 1, 23, 3] for similar results in related settings). If we assume the existence of (families of) collision-resistant hash functions, then there even exist constant-round interactive proof systems that achieve a bounded version of Definition 7 in which

the number of concurrent instances is fixed in advance Barak [1], Barak and Goldreich [2]).[2]

Black-box reductions. Reingold et al. [43] give an excellent overview and classification of black-box reductions. We recall some of their definitions which are important for our case. A *primitive* $\mathsf{P} = (F_{\mathsf{P}}, R_{\mathsf{P}})$ is a set F_{P} of functions $f : \{0,1\}^* \to \{0,1\}^*$ along with a relation R over pairs (f, A), where $f \in F_{\mathsf{P}}$, and A is a machine. We say that f is an *implementation* of P iff $f \in F_{\mathsf{P}}$. Furthermore, f is an *efficient implementation* of P iff $f \in F_{\mathsf{P}}$ and f can be computed by a PPT machine. A machine A P*-breaks* $f \in F_{\mathsf{P}}$ iff $R_{\mathsf{P}}(f, A)$. A primitive P *exists* if there is an efficient implementation $f \in F_{\mathsf{P}}$ such that no PPT machine P-breaks f. A primitive P *exists relative to an oracle* $\mathcal{B}$ iff there exists an implementation $f \in F_{\mathsf{P}}$ which is computable by a PPT machine with access to $\mathcal{B}$, such that no PPT machine with access to $\mathcal{B}$ P-breaks f.

Definition 8 (Relativizing reduction). *There exists a* relativizing reduction *from a primitive* $\mathsf{P} = (F_{\mathsf{P}}, R_{\mathsf{P}})$ *to a primitive* $\mathsf{Q} = (F_{\mathsf{Q}}, R_{\mathsf{Q}})$ *iff for every oracle* $\mathcal{B}$*, the following holds: if* Q *exists relative to* $\mathcal{B}$*, then so does* P*.*

Definition 9 (∀∃semi-black-box reduction). *There exists a* ∀∃semi-black-box reduction *from a primitive* $\mathsf{P} = (F_{\mathsf{P}}, R_{\mathsf{P}})$ *to a primitive* $\mathsf{Q} = (F_{\mathsf{Q}}, R_{\mathsf{Q}})$ *iff for every implementation* $f \in F_{\mathsf{Q}}$*, there exists a PPT machine* G *such that* $G^f \in F_{\mathsf{P}}$*, and the following holds: if there exists a PPT machine* A *such that* A^f P*-breaks* G^f*, then there exists a PPT machine* S *such that* S^f Q*-breaks* f*.*

It can be seen that if a relativizing reduction exists, then so does a ∀∃semi-black-box reduction. The converse is true when Q "allows embedding," which essentially means that additional oracles can be embedded into Q without destroying its functionality (see Reingold et al. [43], Definition 3.4 and Theorem 3.5 and Simon [46]). Below we will prove impossibility of relativizing reductions between certain primitives, which also proves impossibility of ∀∃semi-black-box reductions, since the corresponding primitives Q allow embedding.

7 Simulation-Based Commitment Security under Selective Openings

Consider the following real security game: adversary A gets, say, n commitments, and then may ask for openings of some of them. The security notion of Dwork et al. [22] requires that for any such A, there exists a simulator S that can approximate A's output. More concretely, for any relation R, we require that $R(\mathbf{m}, out_A)$ holds about as often as $R(\mathbf{m}, out_S)$, where $\mathbf{m} = (\mathbf{m}[i])_{i\in[n]}$ are the messages in the commitments, out_A is A's output, and out_S is S's output.

[2] It is common to allow the simulator S^* to be *expected polynomial-time.* In fact, the positive results [44, 33] (but not [1]) construct an expected PPT S^*. We will neglect this issue in the following, since our results do not depend the complexity of S^* (as long as S^* is not able to break an underlying computational assumption).

Formally, we get the following definition (where henceforth, $\mathcal{I}$ will denote the set of "allowed" opening sets):

Definition 10 (SEM-SO-COM). *Assume $n = n(\lambda) > 0$ is polynomially bounded, and let $\mathcal{I} = (\mathcal{I}_n)_n$ be a family of sets such that each $\mathcal{I}_n$ is a set of subsets of $[n]$. A commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ is* simulatable under selective openings *(short* SEM-SO-COM secure*) iff for every PPT n-message distribution $\mathcal{M}$, every PPT relation R, and every PPT machine A (the adversary), there is a PPT machine S (the simulator), such that $\mathbf{Adv}^{\mathsf{sem\text{-}so}}_{\mathsf{Com},\mathcal{M},A,S,R}$ is negligible. Here*

$$\mathbf{Adv}^{\mathsf{sem\text{-}so}}_{\mathsf{Com},\mathcal{M},A,S,R}(\lambda) := \Pr\left[\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}real}}_{\mathsf{Com},\mathcal{M},A,R} = 1\right](\lambda) - \Pr\left[\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}ideal}}_{\mathcal{M},S,R} = 1\right](\lambda),$$

where the experiments $\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}real}}_{\mathsf{Com},\mathcal{M},A,R}$ and $\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}ideal}}_{\mathcal{M},S,R}$ are defined as follows:

Experiment $\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}real}}_{\mathsf{Com},\mathcal{M},A,R}(\lambda)$	**Experiment** $\mathbf{Exp}^{\mathsf{sem\text{-}so\text{-}ideal}}_{\mathcal{M},S,R}(\lambda)$
$\mathbf{m} = (\mathbf{m}[i])_{i\in[n]} \leftarrow\!\!\$\ \mathcal{M}$	$\mathbf{m} = (\mathbf{m}[i])_{i\in[n]} \leftarrow\!\!\$\ \mathcal{M}$
$I \leftarrow\!\!\$\ \langle A(\mathtt{recv}), (\mathsf{S}_i(\mathtt{com}, \mathbf{m}[i]))_{i\in[n]}\rangle$	$I \leftarrow\!\!\$\ S(\mathtt{choose})$
$out_A \leftarrow\!\!\$\ \langle A(\mathtt{open}), (\mathsf{S}_i(\mathtt{open}))_{i\in I}\rangle$	$out_S \leftarrow\!\!\$\ S((\mathbf{m}[i])_{i\in I})$
return $R(\mathbf{m}, out_A)$	return $R(\mathbf{m}, out_S)$

In this, we require from A that $I \in \mathcal{I}_\lambda$,[3] *and we denote by $\langle A, (\mathsf{S}_i)_i\rangle$ the output of A after interacting concurrently with instances S_i of S.*

Discussion of the definitional choices. While Definition 10 essentially is the selective decommitment definition Dwork et al. [22], Definition 7.1, there are a number of definitional choices we would like to highlight (the following discussion applies equally to the upcoming Definition 13):

- Unlike [22, Definition 7.1], neither adversary A nor relation R get an auxiliary input. Such an auxiliary input is common in cryptographic definitions to ensure some form of composability.
- We do not explicitly hand the chosen set I to the relation R. Handing I to R potentially makes the definition more useful in larger contexts in which I is public.
- One could think of letting R determine the message vector $\mathbf{m}$.[4] (Equivalently, we can view $\mathcal{M}$ as part of R and let $\mathcal{M}$ forward its random coins—or a short seed—to R in a message part $\mathbf{m}[i]$ which is guaranteed not to be opened, e.g., when $i \notin I$ for all $I \in \mathcal{I}_n$.)
- The order of quantifiers $(\forall\mathcal{M}, R, A\exists S)$ is the weakest one possible. In particular, we do not mandate that S is constructed from A in a black-box way.

[3] that is, we actually only quantify over those A for which $I \in \mathcal{I}_\lambda$.

[4] This definition is closer to a universally composable definition (cf. Canetti [11]) in the sense that R (almost) takes the role of a UC-environment: R selects all inputs and reads the outputs (in particular the output of A). However, we stress that R may not actively interfere in the commitment protocol. Note that we cannot hope for *fully* UC-secure commitments for reasons not connected to the selective decommitment problem, cf. Canetti and Fischlin [12].

In all of the cases, we chose the weaker definitional variant for simplicity, which makes our negative results only stronger. We stress, however, that our positive results (Theorem 4 and Theorem 6) hold also for all of the stronger definitional variants.

7.1 Impossibility from Black-Box Reductions

Formalization of computational assumptions. Our first negative result states that SEM-SO-COM security cannot be achieved via black-box reductions from standard assumptions. We want to consider such standard assumptions in a general way that allows to make statements even in the presence of "relativizing" oracles. Thus we make the following definition, which is a special case of the definition of a *primitive* from Reingold et al. [43] (cf. also Section 6).

Definition 11 (Property of an oracle). *Let $\mathcal{X}$ be an oracle. Then a* property *$\mathcal{P}$ of $\mathcal{X}$ is a (not necessarily PPT) machine that, after interacting with $\mathcal{X}$ and another machine A, finally outputs a bit b. For an adversary A (that may interact with $\mathcal{X}$ and $\mathcal{P}$), we define A's* advantage against *$\mathcal{P}$ as*

$$\mathbf{Adv}^{\mathsf{prop}}_{\mathcal{P},\mathcal{X},A} := \Pr[\mathcal{P} \text{ outputs } b = 1 \text{ after interacting with } A \text{ and } \mathcal{X}] - 1/2.$$

Now $\mathcal{X}$ is said to satisfy *property $\mathcal{P}$ iff for all PPT adversaries A, we have that $\mathbf{Adv}^{\mathsf{prop}}_{\mathcal{P},\mathcal{X},A}$ is negligible.*

In terms of Reingold et al. [43], the corresponding primitive is $\mathsf{P} = (F_{\mathsf{P}}, R_{\mathsf{P}})$, where $F_{\mathsf{P}} = \{\mathcal{X}\}$, and $R_{\mathsf{P}}(\mathcal{X}, A)$ iff $\mathbf{Adv}^{\mathsf{prop}}_{\mathcal{P},\mathcal{X},A}$ is non-negligible. Our definition is also similar in spirit to "hard games" as used by Dodis et al. [20], but more general.

We emphasize that $\mathcal{P}$ can *only* interact with $\mathcal{X}$ and A, but not with possible additional oracles. (See Section 9 for further discussion of properties of oracles, in particular their role in our proofs.) Intuitively, $\mathcal{P}$ acts as a challenger in the sense of a cryptographic security experiment. That is, $\mathcal{P}$ tests whether adversary A can "break" $\mathcal{X}$ in the intended way. We give an example, where "breaking" means "breaking $\mathcal{X}$'s one-way property".

Example. If $\mathcal{X}$ is a random permutation of $\{0,1\}^\lambda$, then the following $\mathcal{P}$ models $\mathcal{X}$'s one-way property: $\mathcal{P}$ acts as a challenger that challenges A to invert a randomly chosen $\mathcal{X}$-image. Concretely, $\mathcal{P}$ initially chooses a random $Y \in \{0,1\}^\lambda$ and sends Y to A. Upon receiving a guess $X \in \{0,1\}^\lambda$ from A, $\mathcal{P}$ checks if $\mathcal{X}(X) = Y$. If yes, then $\mathcal{P}$ terminates with output $b = 1$. If $\mathcal{X}(X) \neq Y$, then $\mathcal{P}$ tosses an unbiased coin $b' \in \{0,1\}$ and terminates with output $b = b'$.

We stress that we only gain generality by demanding that $\Pr[\mathcal{P}$ outputs $1]$ is close to $1/2$ (and not, say, negligible). In fact, this way indistinguishability-based games (such as, e.g., the indistinguishability of ciphertexts of an ideal encryption scheme $\mathcal{X}$) can be formalized very conveniently. On the other hand, cryptographic games like the one-way game above can be formulated in this framework as well, by letting the challenger output $b = 1$ with probability $1/2$ when A fails.

On the role of property $\mathcal{P}$. Our upcoming results state the impossibility of (black-box) security reductions, from essentially *any* computational assumption (i.e., property) $\mathcal{P}$. The obvious question is: what if the assumption already *is* an idealized commitment scheme secure under selective openings? The short answer is: "then the security proof will not be black-box." We give a detailed explanation of what is going on in Section 9.

Stateless breaking oracles. In our impossibility results, we will describe a computational world with a number of oracles. For instance, there will be a "breaking oracle" $\mathcal{B}$, such that $\mathcal{B}$ aids in breaking the SEM-SO-COM security of any given commitment scheme, and in *nothing more.* To this end, $\mathcal{B}$ takes the role of the adversary in the SEM-SO-COM experiment. Namely, $\mathcal{B}$ expects to receive a number of commitments, then chooses a subset of these commitments, and then expects openings of the commitments in this subset. This is an interactive process which would usually require $\mathcal{B}$ to hold a state across invocations. However, stateful oracles are not very useful for establishing black-box separations, so we will have to give a stateless formulation of $\mathcal{B}$. Concretely, suppose that the investigated commitment scheme is non-interactive. Then $\mathcal{B}$ answers deterministically upon queries and expects each query to be prefixed with the history of that query. For instance, $\mathcal{B}$ finally expects to receive openings $dec = (dec[i])_{i\in I}$ *along* with the corresponding previous commitments $com = (com[i])_{i\in[n]}$ and previously selected set I. If I is not the set that $\mathcal{B}$ would have selected when receiving com alone, then $\mathcal{B}$ ignores the query. This way, $\mathcal{B}$ is stateless (but randomized, similarly to a random oracle). Furthermore, for non-interactive commitment schemes, this makes sure that any machine interacting with $\mathcal{B}$ can open commitments to $\mathcal{B}$ only in one way. Hence this formalization preserves the binding property of a commitment scheme, something which we will need in our proofs.

We stress, however, that this method does not necessarily work for interactive commitment schemes. Namely, any machine interacting with such a stateless $\mathcal{B}$ can essentially "rewind" $\mathcal{B}$ during an interactive commitment phase, since $\mathcal{B}$ formalizes a next-message function. Now if the commitment scheme is still binding if the receiver of the commitment can be rewound (e.g., this holds trivially for non-interactive commitment schemes, and also for perfectly binding commitment schemes), then our formalization of $\mathcal{B}$ preserves binding, and our upcoming proof works. If, however, the commitment scheme loses its binding property if the receiver can be rewound, then the following theorem cannot be applied.

We are now ready to state our result.

Theorem 3 (Black-box impossibility of non-interactive or perfectly binding SEM-SO-COM, most general formulation). *Let $n = n(\lambda) = 2\lambda$, and let $\mathcal{I} = (\mathcal{I}_n)_n$ with $\mathcal{I}_n = \{I \subseteq [n] \mid |I| = n/2\}$ denote the set of all $n/2$-sized subsets of $[n]$.*[5] *Let $\mathcal{X}$ be an oracle that satisfies property $\mathcal{P}$. Then there is a set of oracles relative to which $\mathcal{X}$ still satisfies property $\mathcal{P}$, but there exists no*

[5] We stress that the proofs of Theorem 3 and Theorem 5 hold literally also for the "cut-and-choose" $\mathcal{I}_n = \{I \subseteq [n] \mid \forall i \in [\lambda] :$ either $2i - 1 \in I$ or $2i \in I\}$.

non-interactive or perfectly binding commitment scheme which is simulatable under selective openings.

Proof strategy. We will use a random oracle $\mathcal{RO}$ that, for any given non-interactive commitment scheme Com^*, induces a message distribution $\mathcal{M}^* = \{(\mathcal{RO}(\mathsf{Com}^*, i, X^*))_{i\in[n]}\}_{X^*\in\{0,1\}^{\lambda/3}}$. Here, $\mathcal{RO}(\mathsf{Com}^*)$ denotes the hash of the description of Com^*, and X^* is a short "seed" that ties the values $\mathcal{RO}(\mathsf{Com}^*, i, X^*)$ (with the same X^* but different i) together. Furthermore, we will specify an oracle $\mathcal{B}$ that will help to break Com^* with respect to $\mathcal{M}^*$. Concretely, $\mathcal{B}$ first expects n Com^*-commitments, and then requests openings of a random subset of them. If all openings are valid, $\mathcal{B}$ returns a value X^* consistent (according to $\mathcal{M}^*$) with all opened messages (if such an X^* exists). A suitable SEM-SO-COM adversary A can use $\mathcal{B}$ simply by relaying its challenge to obtain X^* and hence the whole message vector in its SEM-SO-COM experiment.

However, we will prove that $\mathcal{B}$ is useless to any simulator S that gets only a message subset $\mathbf{m}[I]$: if S uses $\mathcal{B}$ *before* requesting its own message subset $\mathbf{m}[I]$, then $\mathcal{B}$'s answer will not be correlated with the SEM-SO-COM challenge message vector $\mathbf{m}$. (This also holds if S first sends commitments to $\mathcal{B}$ and immediately afterwards requests $\mathbf{m}[I]$ from the SEM-SO-COM experiment; in that case, S has to break the binding property of Com^* to get an answer from $\mathcal{B}$ which is correlated with $\mathbf{m}$.) But if S uses $\mathcal{B}$ *after* obtaining $\mathbf{m}[I]$, then with very high probability, S will have open at least one commitment to $\mathcal{B}$ whose message is not contained in $\mathbf{m}[I]$. By definition of $\mathcal{M}^*$, this opening of S will not be consistent with the other values of $\mathbf{m}[I]$ (except with small probability), and $\mathcal{B}$'s answer will again not be correlated with $\mathbf{m}$.

Since S cannot efficiently extract the seed X^* from its message subset $\mathbf{m}[I]$ alone (that would require a brute-force search over exponentially many values), this shows that Com^* is not SEM-SO-COM secure. Consequently, because Com^* was arbitrary (only the message distribution $\mathcal{M}^*$ is specific to Com^*), there exist no SEM-SO-COM secure commitment schemes relative to $\mathcal{RO}$ and $\mathcal{B}$. Finally, it is easy to see that relative to $\mathcal{RO}$ and $\mathcal{B}$, primitive $\mathcal{X}$ still satisfies property $\mathcal{P}$. Concretely, observe that $\mathcal{B}$ does not break any commitment (note that $\mathcal{B}$'s answer depends only on the *opened* commitments), but only inverts a message distribution (or, rather, $\mathcal{RO}$). Hence, any adversary attacking property $\mathcal{P}$ of $\mathcal{X}$ can use efficient internal simulations of $\mathcal{RO}$ and $\mathcal{B}$ instead of the original oracles. Since $\mathcal{X}$ satisfies property $\mathcal{P}$ with respect to adversaries without (additional) oracle access, the claim follows.

The following corollary provides an instantiation of Theorem 3 for a number of standard cryptographic primitives.

Corollary 1 (Black-box impossibility of non-interactive or perfectly binding SEM-SO-COM). *Assume n and $\mathcal{I}$ as in Theorem 3. Then no non-interactive or perfectly binding commitment scheme can be proved simulatable under selective openings via a $\forall\exists$semi-black-box reduction to one or more of the following primitives: one-way functions, one-way permutations, trapdoor one-way permutations, IND-CCA secure public key encryption, homomorphic public key encryption.*

The corollary is a special case of Theorem 3. For instance, to show Corollary 1 for one-way permutations, one can use the example $\mathcal{X}$ and $\mathcal{P}$ from above: $\mathcal{X}$ is a random permutation of $\{0,1\}^\lambda$, and $\mathcal{P}$ models the one-way experiment with $\mathcal{X}$. Clearly, $\mathcal{X}$ satisfies $\mathcal{P}$, and so we can apply Corollary 1. This yields impossibility of relativizing proofs for SEM-SO-COM security from one-way permutations. We get impossibility for $\forall\exists$semi-black-box reductions since one-way permutations allow embedding, cf. Simon [46],Reingold et al. [43]. The other cases are similar. Note that while it is generally not easy to even give a candidate for a cryptographic primitive in the standard model, it is easy to construct an idealized, say, encryption scheme in oracle form.

We stress that Corollary 1 makes no assumptions about the nature of the simulation (in the sense of Definition 10). In particular, the simulator may freely use, e.g., the code of the adversary; the only restriction is black-box access to the underlying primitive. As discussed in the introduction, this is quite different from the result one gets upon combining Goldreich and Krawczyk [26] and Dwork et al. [22]: essentially, combining [26, 22] shows impossibility of constructing S in a black-box way from A (i.e., such that S only gets black-box access to A's next-message function).

Generalizations. First, Corollary 1 constitutes merely an example instantiation of the much more general Theorem 3. Second, the proof also holds for a relaxation of SEM-SO-COM security considered by Dwork et al. [22], Definition 7.3, where adversary and simulator approximate a function of the message vector.

7.2 Possibility Using Non-black-box Techniques

Non-black-box techniques vs. interaction. Theorem 3 shows that SEM-SO-COM security cannot be achieved unless one uses non-black-box techniques or interaction. In this section, we will investigate the power of non-black-box techniques to achieve SEM-SO-COM security. As it turns out, for our purposes a concurrently composable zero-knowledge argument system is a suitable non-black-box tool.[6] We stress that the use of this zero-knowledge argument makes our scheme necessarily interactive, and so actually circumvents Theorem 3 in *two* ways: by non-black-box techniques *and* by interaction. However, from a conceptual point of view, our scheme is "non-interactive up to the zero-knowledge argument." In particular, our proof does not use the fact that the zero-knowledge argument is interactive. (That is, if we used a concurrently composable non-interactive zero-knowledge argument in, say, the common reference string model, our proof would still work.)

The scheme. For our non-black-box scheme, we need an interactive argument system IP with perfect completeness and negligible soundness error, such that IP is zero-knowledge under concurrent composition. We also need a perfectly binding non-interactive commitment scheme $\mathsf{Com}^{\mathsf{b}}$. Both these ingredients can be

[6] We require concurrent composability since the SEM-SO-COM definition considers multiple, concurrent sessions of the commitment scheme.

constructed from one-way permutations. To ease presentation, we only describe a *bit* commitment scheme, which is easily extended (along with the proof) to the multi-bit case. In a nutshell, the sender S^{ZK} commits twice (using $\mathsf{Com}^{\mathsf{b}}$) to the the same bit and proves in zero-knowledge (using IP) that the committed bits are the same.[7] In the opening phase, the sender opens one (randomly selected) commitment. Note that this overall commitment scheme is binding, since IP ensures that both commitments contain the same bits, and the underlying commitment $\mathsf{Com}^{\mathsf{b}}$ is binding. For a SEM-SO-COM simulation, we generate inconsistent overall commitments which can later be opened arbitrarily by choosing which individual $\mathsf{Com}^{\mathsf{b}}$-commitment is opened. We can use the simulator of IP to generate fake consistency proofs for these inconsistent commitments. (Since we consider many concurrent commitment instances in our SEM-SO-COM experiment, we require concurrent composability from IP for that.)

Scheme 12 (Non-black-box commitment scheme ZKCom). Let $\mathsf{Com}^{\mathsf{b}} = (\mathsf{S}^{\mathsf{b}}, \mathsf{R}^{\mathsf{b}})$ be a perfectly binding non-interactive commitment scheme. Let $\mathsf{IP} = (\mathsf{P}, \mathsf{V})$ be an interactive argument system for NP which enjoys perfect completeness, has negligible soundness error, and which is zero-knowledge under concurrent composition. Let $\mathsf{ZKCom} = (\mathsf{S}^{\mathsf{ZK}}, \mathsf{R}^{\mathsf{ZK}})$ for the following S^{ZK} and R^{ZK}:

- Commitment to bit b:
  1. S^{ZK} prepares $(com^j, dec^j) \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{b}}(b)$ for $j \in \{0,1\}$ and sends (com^0, com^1) to R^{ZK}.
  2. S^{ZK} uses IP to prove to R^{ZK} that com^0 and com^1 commit to the same bit.[8]
- Opening:
  1. S^{ZK} uniformly chooses $j \in \{0,1\}$ and sends (j, dec^j) to R^{ZK}.

The security of ZKCom. It is straightforward to prove that ZKCom is a hiding and binding commitment scheme. (We stress, however, that $\mathsf{Com}^{\mathsf{b}}$'s *perfect* binding property is needed to prove that ZKCom is binding; otherwise, the zero-knowledge argument may become meaningless.) More interestingly, we can also show that ZKCom is SEM-SO-COM secure:

Theorem 4 (Non-black-box possibility of SEM-SO-COM). *Fix n and $\mathcal{I}$ as in Definition 10. Then* ZKCom *is simulatable under selective openings in the sense of Definition 10.*

[7] We note that a FOCS referee, reviewing an earlier version of this paper without ZKCom, also suggested to employ zero-knowledge to prove consistency of a given commitment. This suggestion was independent of the eprint version of this paper which at that time already contained our scheme ZKCom. A Eurocrypt referee, reviewing a version of the paper with ZKCom, remarked that alternative constructions of a SEM-SO-COM secure commitment scheme are possible. A more generic construction could be along the lines of "commit using a perfectly binding commitment, then prove consistency of commitment or opening using concurrent zero-knowledge."

[8] Formally, the corresponding language $\mathcal{L}$ for IP consists of statements $x = (com^0, com^1)$ and witnesses $w = (dec^0, dec^1)$ such that $\mathcal{R}(x,w)$ iff $\mathsf{R}^{\mathsf{b}}(com^0, dec^0) = \mathsf{R}^{\mathsf{b}}(com^1, dec^1) \in \{0,1\}$.

Proof outline. We start with the real SEM-SO-COM experiment with an arbitrary adversary A. As a first step, we substitute the proofs generated during the commitments by *simulated proofs*. Concretely, we hand to A proofs for the consistency of the commitments that are generated by a suitable simulator S^*. By the concurrent zero-knowledge property of IP, such an S^* exists and yields indistinguishable experiment outputs. Note that S^* does not need witnesses to generate valid-looking proofs, but instead uses (possibly rewinding or even non-black-box) access to A. Hence, we can substitute all ZKCom-commitments with inconsistent commitments of the form (com^0, com^1), where com^0 and com^1 are Com^b-commitments to *different* bits. Such a ZKCom-commitment can later be opened arbitrarily. By the computational hiding property of Com^b (and since we do not need witnesses to generate consistency proofs anymore), this step does not change the output distribution of the experiment significantly. But note that now, the initial generation of the commitments does not need knowledge of the actual messages. In fact, only the messages $\mathbf{m}[I]$ of the actually opened commitments need to be known at opening time. Hence, at this point, the modified experiment is a valid simulator in the sense of the ideal SEM-SO-COM experiment. Since the experiment output has only been changed negligibly by our modifications, we have thus constructed a successful simulator in the sense of Definition 10.

Where is the non-black-box component? Interestingly, the used argument system IP itself can well be black-box zero-knowledge (where black-box zero-knowledge means that the simulator S^* from Definition 7 has only black-box access to the next-message function of V^*). The essential fact that allows us to circumvent our negative result Theorem 3 is the way we employ IP. Namely, ZKCom uses IP to prove a statement about two given commitments (com^0, com^1). This proof (or, rather, argument) uses an explicit and non-black-box description of the employed commitment scheme Com^b. It is this argument that cannot even be expressed when Com^b makes use of, say, a one-way function given in oracle form.

The role of the commitment randomness. Observe that the opening of a ZKCom-commitment does not release all randomness used for constructing the commitment. In fact, it is easy to see that our proof would not hold if S^{ZK} opened *both* commitments com^0 and com^1 in the opening phase. Hence, ZKCom is not suitable for settings in which an opening corresponds to a corruption of a party (e.g., in a multi-party computation setting), and when one cannot assume no trusted erasures.

Generalizations. First, ZKCom can be straightforwardly extended to a multi-bit commitment scheme, e.g., by running several sessions of ZKCom in parallel. Second, ZKCom is SEM-SO-COM secure also against adversaries with auxiliary input z: our proof holds literally, where of course we also require security of Com^b against non-uniform adversaries.

8 Indistinguishability-Based Commitment Security under Selective Openings

Motivated by the impossibility result from the previous section, we now relax Definition 10 as follows:

Definition 13 (IND-SO-COM). *Let $n = n(\lambda) > 0$ be polynomially bounded, and let $\mathcal{I} = (\mathcal{I}_n)_n$ be a family of sets such that each $\mathcal{I}_n$ is a set of subsets of $[n]$. A commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ is* indistinguishable under selective openings *(short* IND-SO-COM secure*) iff for every PPT n-message distribution $\mathcal{M}$, and every PPT adversary A, we have that $\mathbf{Adv}^{\mathsf{ind\text{-}so}}_{\mathsf{Com},\mathcal{M},A}$ is negligible. Here*

$$\mathbf{Adv}^{\mathsf{ind\text{-}so}}_{\mathsf{Com},\mathcal{M},A}(\lambda) := \Pr\left[\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}real}}_{\mathsf{Com},\mathcal{M},A} = 1\right](\lambda) - \Pr\left[\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}ideal}}_{\mathsf{Com},\mathcal{M},A} = 1\right](\lambda),$$

where the experiments $\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}real}}_{\mathsf{Com},\mathcal{M},A}$ and $\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}ideal}}_{\mathsf{Com},\mathcal{M},A}$ are defined as follows:

Experiment $\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}real}}_{\mathsf{Com},\mathcal{M},A}(\lambda)$	**Experiment** $\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}ideal}}_{\mathsf{Com},\mathcal{M},A}(\lambda)$
$\mathbf{m} = (\mathbf{m}[i])_{i\in[n]} \leftarrow\!\!\$\ \mathcal{M}$	$\mathbf{m} = (\mathbf{m}[i])_{i\in[n]} \leftarrow\!\!\$\ \mathcal{M}$
$I \leftarrow\!\!\$\ \langle A(\mathtt{recv}), (\mathsf{S}_i(\mathtt{com}, \mathbf{m}[i]))_{i\in[n]}\rangle$	$I \leftarrow\!\!\$\ \langle A(\mathtt{recv}), (\mathsf{S}_i(\mathtt{com}, \mathbf{m}[i]))_{i\in[n]}\rangle$
$out_A \leftarrow\!\!\$\ \langle A(\mathtt{open}), (\mathsf{S}_i(\mathtt{open}))_{i\in I}\rangle$	$out_A \leftarrow\!\!\$\ \langle A(\mathtt{open}), (\mathsf{S}_i(\mathtt{open}))_{i\in I}\rangle$
	$\mathbf{m}' \leftarrow\!\!\$\ \mathcal{M} \mid \mathbf{m}[I]$
return $A(\mathtt{guess}, \mathbf{m})$	return $A(\mathtt{guess}, \mathbf{m}')$

Again, we require from A that $I \in \mathcal{I}_\lambda$, and we denote by $\langle A, (\mathsf{S}_i)_i\rangle$ the output of A after interacting concurrently with instances S_i of S. Furthermore, $\mathcal{M} \mid \mathbf{m}[I]$ denotes the message distribution $\mathcal{M}$ conditioned on the values of $\mathbf{m}[I]$.

On the conditioned distribution $\mathcal{M} \mid \mathbf{m}[I]$. We stress that, depending on $\mathcal{M}$, it may be computationally hard to sample $\mathbf{m}' \leftarrow\!\!\$\ \mathcal{M} \mid \mathbf{m}[I]$, even if (the unconditioned) $\mathcal{M}$ is PPT. This might seem strange at first and inconvenient when *applying* the definition in some larger reduction proof. However, there simply seems to be no other way to capture indistinguishability, since the set of opened commitments depends on the commitments themselves. In particular, in general we cannot predict which commitments the adversary wants opened, and then, say, substitute the not-to-be-opened commitments with random commitments. What we chose to do instead is to give the adversary either the full message vector, or an independent message vector which "could be" the full message vector, given the opened commitments. We believe that this is the canonical way to capture secrecy of the unopened commitments under selective openings.

The relation between SEM-SO-COM and IND-SO-COM security. Unfortunately, we (currently) cannot prove that SEM-SO-COM security implies IND-SO-COM security (although this seems plausible, since usually simulation-based definitions imply their indistinguishability-based counterparts). Technically, the reason why we are unable to prove an implication is the conditioned distribution $\mathcal{M} \mid \mathbf{m}[I]$ in the ideal IND-SO-COM experiment, which cannot be sampled from during an (efficient) reduction.

A relaxation. Alternatively, we could let the adversary predict a predicate π of the whole message vector, and consider him successful if $\Pr[b = \pi(\mathbf{m})]$ and $\Pr[b = \pi(\mathbf{m}')]$ for the alternative message vector $\mathbf{m}' \leftarrow\!\!\$\ \mathcal{M} \mid \mathbf{m}[I]$ differ non-negligibly. We stress that our upcoming negative result also applies to this relaxed notion.

8.1 Impossibility from Black-Box Reductions

Theorem 5 (Black-box impossibility of perfectly binding IND-SO-COM, most general formulation). *Let $n = n(\lambda) = 2\lambda$, and let $\mathcal{I} = (\mathcal{I}_n)_n$ with $\mathcal{I}_n = \{I \subseteq [n] \mid |I| = n/2\}$ denote the set of all $n/2$-sized subsets of $[n]$. Let $\mathcal{X}$ be an oracle that satisfies a property $\mathcal{P}$ even in presence of an EXPSPACE-oracle. We also assume that $\mathcal{X}$ is computable in EXPSPACE.*[9] *Then, there exists a set of oracles relative to which $\mathcal{X}$ still satisfies $\mathcal{P}$, but no perfectly binding commitment scheme is indistinguishable under selective openings.*

Proof outline. Similarly to Theorem 3, we specify an oracle $\mathcal{RO}$ which induces a message distribution $\mathcal{M}^*$. This time, however, $\mathcal{RO}$ maps $\mathbb{E}^{n/2+1}$-elements to message vectors in $\mathbb{E}^n$, where $\mathbb{E} = \{0,1\}^\lambda$ is the domain of each individual message. Hence, $n/2$ messages usually do not fix the whole message vector, but more messages do. Now fix any perfectly binding commitment scheme Com^*. We define a breaking oracle $\mathcal{B}$ that, like the $\mathcal{B}$ from Theorem 3, asks for n Com^*-commitments and subsequent openings of a random subset $I \in \mathcal{I}_n$ of these commitments. If all openings are valid, $\mathcal{B}$ extracts the *whole* message vector in the commitments (note that this is possible since Com^* is perfectly binding), and returns a "close" (with respect to Hamming distance) element in the message distribution $\mathcal{M}^*$ if there is a sufficiently close one.

It is easy to see that an adversary can use $\mathcal{B}$ to obtain the whole message vector $\mathbf{m}$ in the real IND-SO-COM experiment. But a message vector freshly sampled from $\mathcal{M}^*$, conditioned on the opened messages $\mathbf{m}[I]$, will most likely be different from $\mathbf{m}$. Hence, our adversary easily distinguishes the real from the ideal IND-SO-COM experiment.

The main part of the proof shows that oracle $\mathcal{B}$ is useless to an adversary attacking $\mathcal{X}$'s property $\mathcal{P}$. Assume first that the commitment scheme Com with respect to which an adversary A on $\mathcal{X}$ queries $\mathcal{B}$ is perfectly binding. In that case, a somewhat technical but straightforward combinatorial argument shows that A's successfully opened messages $\mathbf{m}[I]$, *together with A's queries to $\mathcal{RO}$,* determine $\mathcal{B}$'s answer (except with small probability). Hence A can use internal simulations of $\mathcal{B}$ and $\mathcal{RO}$ instead of the original oracles, and hence property $\mathcal{P}$ of $\mathcal{X}$ is not damaged by the presence of $\mathcal{B}$. To ensure that $\mathcal{B}$ is only useful for perfectly binding commitment schemes Com, we let $\mathcal{B}$ *test* whether Com is perfectly binding. Since we demand that Com is *perfectly* binding, this test is independent of the random coins used by $\mathcal{X}$. Indeed, $\mathcal{B}$ needs to check that for all syntactically possible commitments and decommitments, and *all* possible

[9] Examples of such $\mathcal{X}$ are random oracles or ideal ciphers. It will become clearer how we use the EXPSPACE requirement in the proof.

random coins used by $\mathcal{X}$, the opened message is unique. Hence, by assumption about $\mathcal{X}$, this test can also be performed by A using an EXPSPACE-oracle, and the above proof idea applies.

On the requirement on $\mathcal{X}$. We stress that the requirement in Theorem 5 on $\mathcal{X}$ is a rather mild one. For instance, random oracles are one-way even against computationally unbounded adversaries, as long as the adversary makes only a polynomial number of oracle queries. Hence, an EXPSPACE-oracle (which itself does not perform oracle queries) is not helpful in breaking a random oracle. So similarly to Corollary 1, we get for concrete choices of $\mathcal{X}$ and $\mathcal{P}$:

Corollary 2 (Black-box impossibility of perfectly binding IND-SO-COM). *Let n and $\mathcal{I}$ as in Theorem 5. Then no perfectly binding commitment scheme can be proved indistinguishable under selective openings via a $\forall\exists$semi-black-box reduction to one or more of the following primitives: one-way functions, one-way permutations, trapdoor one-way permutations, IND-CCA secure public key encryption, homomorphic public key encryption.*

Generalizations. Again, Corollary 2 constitutes merely an example instantiation of the much more general Theorem 5. We stress, however, that the proof for Theorem 5 does *not* apply to "almost-perfectly binding" commitment schemes such as the one from Naor [35]. (For instance, for such schemes, $\mathcal{B}$'s check that the supplied commitment scheme is binding might tell something about $\mathcal{X}$.)

8.2 Statistically Hiding Schemes Are Secure

Fortunately, things look different for statistically hiding commitment schemes:

Theorem 6 (Statistically hiding schemes are IND-SO-COM secure). *Fix arbitrary n and $\mathcal{I}$ as in Definition 13, and let $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ be a statistically hiding commitment scheme. Then Com is indistinguishable under selective openings in the sense of Definition 13.*

Proof outline. Intuitively, the claim holds since an adversary A's views in the real, resp. ideal IND-SO-COM experiment are statistically close (and hence so must be A's outputs). However, the fact that A's views are indeed statistically close is less obvious than it may seem at first glance. Our proof proceeds in games and starts with the real IND-SO-COM experiment with A. As a first modification, we change the opening phase of the experiment, so that the opening of each selected commitment is produced solely from the commitment itself and the "target message" $\mathbf{m}[i]$ to which it should be opened (but not from opening information previously generated alongside the commitment). Note that this change is merely conceptual and does not alter A's view at all. This makes the opening phase inefficient, but since we are dealing with statistically hiding commitment schemes, we need not worry about that. Indeed, by the statistical hiding property, we can now substitute all commitments (in a hybrid argument) with commitments to a fixed value (say, 0^λ) without affecting the experiment

output. We can reduce this step to the hiding property of the commitment scheme since the experiment only needs commitments as input, and produces all openings on its own. At this point, all commitments that A gets are independent of $\mathbf{m}$, and so the whole view of A is independent of the unopened values $\mathbf{m}[[n] \setminus I]$. Hence A's output is (almost) independent of $\mathbf{m}[[n] \setminus I]$ in the real IND-SO-COM experiment and, with similar reasoning, also in the ideal IND-SO-COM experiment. This shows the claim.

9 On the Role of Property $\mathcal{P}$

The intuitive contradiction. The formulations of Theorem 3 and Theorem 5 seem intuitively much too general: essentially they claim impossibility of black-box proofs from *any* computational assumption which is formulated as a property $\mathcal{P}$ of an oracle $\mathcal{X}$. Why can't we choose $\mathcal{X}$ to be an ideally secure commitment scheme, and $\mathcal{P}$ a property that models precisely what we want to achieve, e.g., Definition 13 (i.e., IND-SO-COM security)? After all, Definition 13 can be rephrased as a property $\mathcal{P}$ by letting A choose a message distribution $\mathcal{M}$ and send this distribution (as a description of a PPT algorithm $\mathcal{M}$) to $\mathcal{P}$. Then, $\mathcal{P}$ could perform the $\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}real}}_{\mathsf{Com},\mathcal{M},A}$ or the $\mathbf{Exp}^{\mathsf{ind\text{-}so\text{-}ideal}}_{\mathsf{Com},\mathcal{M},A}$ experiment with A, depending on an internal coin toss (the output of $\mathcal{P}$ will then depend on A's output and on that coin toss). This $\mathcal{P}$ models Definition 13, in the sense that

$$\mathbf{Adv}^{\mathsf{ind\text{-}so}}_{\mathsf{Com},\mathcal{M},A} = 2\mathbf{Adv}^{\mathsf{prop}}_{\mathcal{P},\mathcal{X},A}.$$

Also, using a truly random permutation as a basis, it is natural to assume that we can construct an *ideal* (i.e., as an oracle) perfectly binding commitment scheme $\mathcal{X}$ that satisfies $\mathcal{P}$. (Note that although $\mathcal{X}$ is perfectly binding, A's view may still be almost statistically independent of the unopened messages, since the scheme $\mathcal{X}$ is given in oracle form.)

Hence, if the assumption essentially *is* already IND-SO-COM security, we can certainly achieve IND-SO-COM security (in particular, using a trivial reduction), and this seems to contradict Theorem 5. So where is the problem?

Resolving the situation. The problem in the above argument is that $\mathcal{P}$-security (our assumption) implies IND-SO-COM security (our goal) in a fundamentally non-black-box way. Namely, the proof converts an IND-SO-COM adversary A and a message distribution $\mathcal{M}$ into a $\mathcal{P}$-adversary A' that sends a description of $\mathcal{M}$ to $\mathcal{P}$. This very step makes use of an *explicit representation* of the message distribution $\mathcal{M}$, and this is what makes the whole proof non-black-box. In other words, this way of achieving IND-SO-COM security cannot be black-box, and there is no contradiction to our results.

Viewed from a different angle, the essence of our impossibility proofs is: build a very specific message distribution, based on oracles ($\mathcal{RO}$, resp. $\mathcal{C}$), such that another "breaking oracle" $\mathcal{B}$ "breaks" this message distribution if and only if the adversary can prove that he can open commitments. This step relies on the fact that we can specify message distributions which depend on oracles. Relative to such oracles, property $\mathcal{P}$ still holds (as we prove), but may not reflect

IND-SO-COM security anymore. Namely, since $\mathcal{P}$ itself cannot access additional oracles[10], $\mathcal{P}$ is also not able to sample a message space that depends on additional (i.e., on top of $\mathcal{X}$) oracles. So in our reduction, although A itself can, both in the IND-SO-COM experiment and when interacting with $\mathcal{P}$, access all oracles, it will not be able to communicate a message distribution $\mathcal{M}$ that depends on additional oracles (on top of $\mathcal{X}$) to $\mathcal{P}$. On the other hand, any PPT algorithm $\mathcal{M}$, as formalized in Definition 13, *can* access all available oracles.

So for the above modeling of IND-SO-COM as a property $\mathcal{P}$ in the sense of Definition 11, our impossibility results still hold, but become meaningless (since basically using property $\mathcal{P}$ makes the proof non-black-box). In a certain sense, this comes from the fact that the modeling of IND-SO-COM as a property $\mathcal{P}$ is inherently non-black-box. A similar argument holds for the message distribution in the SEM-SO-COM experiment; there, however, we face the additional problem of modeling the existence of a simulator in a property.

What computational assumptions can be formalized as properties in a "black-box" way? Fortunately, most standard computational assumptions can be modeled in a black-box way as a property $\mathcal{P}$. Besides the mentioned one-way property (and its variants), in particular, e.g., the IND-CCA security game for encryption schemes can be modeled. Observe that in this game, we can let the IND-CCA adversary himself sample challenge messages m_0, m_1 for the IND-CCA experiment from his favorite distribution; no PPT algorithm has to be transported to the security game. In fact, the only properties which do not allow for black-box proofs are those that involve an explicit transmission of code (i.e., a description of a circuit or a Turing machine). In that sense, the formulation of Theorem 3 and Theorem 5 is very general and useful.

(Non-)programmable random oracles. We stress that the black-box requirement for random oracles (when used in the role of $\mathcal{X}$) corresponds to "non-programmable random oracles" (as used by, e.g., Bellare and Rogaway [5]) as opposed to "programmable random oracles" (as used by, e.g., Nielsen [38]). Roughly, a proof in the programmable random oracle model translates an attack on a cryptographic scheme into an attack on a *simulated* random oracle (that is, an oracle completely under control of simulator). Naturally, such a reduction is not black-box. And indeed, with programmable random oracles, even non-interactive SEM-SO-COM secure commitment schemes can be built relatively painlessly. As an example, [38] proves a simple encryption scheme (which can be interpreted as a non-interactive commitment scheme) secure under selective openings.

Acknowledgements

Bellare and Yilek thank Saurabh Panjwani for participating in early stages of this work, which involved the development of the indistinguishability-based definition

[10] by Definition 11, $\mathcal{P}$ must be specified independently of additional oracles; if we did allow $\mathcal{P}$ to access additional oracles, this would break our impossibility proofs

IND-SO-ENC. Hofheinz would like to thank Enav Weinreb, Marc Stevens, Serge Fehr, Krzysztof Pietrzak, and Ivan Damgård for many insightful discussions.

Mihir Bellare is supported by NSF grants CNS–0524765 and CNS–0627779 and a gift from Intel Corporation. Dennis Hofheinz is supported by NWO. Scott Yilek is supported by NSF grants CNS–0430595 and CNS–0831536.

References

[1] Barak, B.: How to go beyond the black-box simulation barrier. In: 42nd Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001, pp. 106–115. IEEE Computer Society, Los Alamitos (2001)

[2] Barak, B., Goldreich, O.: Universal arguments and their applications. In: 17th Annual IEEE Conference on Computational Complexity, Proceedings of CoCo 2002, pp. 194–203. IEEE Computer Society, Los Alamitos (2002)

[3] Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero-knowledge. In: 47th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2006, pp. 345–354. IEEE Computer Society, Los Alamitos (2006)

[4] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: 1st ACM Conference on Computer and Communications Security, Proceedings of CCS 1993, pp. 62–73. ACM Press, New York (1993)

[5] Bellare, M., Rogaway, P.: Optimal asymmetric encryption—how to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)

[6] Bellare, M., Rogaway, P.: Robust computational secrete sharing and a unified account of classical secret-sharing goals. In: 14th ACM Conference on Computer and Communications Security, Proceedings of CCS 2007, pp. 172–184. ACM Press, New York (2007)

[7] Bellare, M., Yilek, S.: Encryption schemes secure under selective opening attack. IACR ePrint Archive (2009)

[8] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: 20th ACM Symposium on Theory of Computing, Proceedings of STOC 1988, pp. 1–10. ACM, New York (1988)

[9] Blum, M.: Coin flipping by telephone. In: Gersho, A. (ed.) Advances in Cryptology, A report on CRYPTO 1981, number 82-04 in ECE Report, pp. 11–15. University of California, Electrical and Computer Engineering (1982)

[10] Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)

[11] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001, pp. 136–145. IEEE Computer Society, Los Alamitos (2001)

[12] Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

[13] Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: Twenty-Eighth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1995, pp. 639–648. ACM Press, New York (1996)

[14] Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)

[15] Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In: 33rd Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2001, pp. 570–579. ACM Press, New York (2001)
[16] Canetti, R., Halevi, S., Katz, J.: Adaptively-secure, non-interactive public-key encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 150–168. Springer, Heidelberg (2005)
[17] Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: 20th ACM Symposium on Theory of Computing, Proceedings of STOC 1988, pp. 11–19. ACM Press, New York (1988)
[18] Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on general complexity assumptions. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
[19] Damgård, I.B., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop sigantures. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 250–265. Springer, Heidelberg (1994)
[20] Dodis, Y., Oliveira, R., Pietrzak, K.: On the generic insecurity of the full domain hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
[21] Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: Twenty-Third Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1991, pp. 542–552. ACM Press, New York (1991) (Extended abstract)
[22] Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.: Magic functions. Journal of the ACM 50(6), 852–921 (2003)
[23] Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. Journal of the ACM 51(6), 851–898 (2004)
[24] Gennaro, R., Micali, S.: Independent zero-knowledge sets. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 34–45. Springer, Heidelberg (2006)
[25] Goldreich, O.: Foundations of Cryptography (Basic Tools), vol. 1. Cambridge University Press, Cambridge (2001)
[26] Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM Journal on Computing 25(1), 169–192 (1996)
[27] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2) (1984)
[28] Haitner, I., Holenstein, T.: On the (im)possibility of key dependent encryption. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 202–219. Springer, Heidelberg (2009)
[29] Haitner, I., Reingold, O.: Statistically-hiding commitment from any one-way function. In: 39th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2007, pp. 1–10. ACM Press, New York (2007)
[30] Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols – a tight lower bound on the round complexity of statistically-hiding commitments. In: 48th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2007, pp. 669–679. IEEE Computer Society, Los Alamitos (2007)
[31] Hofheinz, D.: Possibility and impossibility results for selective decommitments. IACR ePrint Archive (April 2008)
[32] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Twenty-First Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1989, pp. 44–61. ACM Press, New York (1989) (Extended abstract)

[33] Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In: 33rd Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2001, pp. 560–569. ACM Press, New York (2001)
[34] Kol, G., Naor, M.: Cryptography and game theory: Designing protocols for exchanging information. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 320–339. Springer, Heidelberg (2008)
[35] Naor, M.: Bit commitment using pseudo-randomness. Journal of Cryptology 4(2), 151–158 (1991)
[36] Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Twelfth Annual Symposium on Discrete Algorithms, Proceedings of SODA 2001, pp. 448–457. ACM/SIAM (2001)
[37] Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Twenty-First Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1989, pp. 33–43. ACM Press, New York (1989)
[38] Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
[39] Panjwani, S.: Tackling adaptive corruptions in multicast encryption protocols. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 21–40. Springer, Heidelberg (2007)
[40] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Fotieth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2008, pp. 187–196. ACM Press, New York (2008)
[41] Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
[42] Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round complexity. In: 43rd Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2002, pp. 366–375. IEEE Computer Society Press, Los Alamitos (2002)
[43] Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
[44] Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–431. Springer, Heidelberg (1999)
[45] Rosen, A., Segev, G.: Efficient lossy trapdoor functions based on the composite residuosity assumption. IACR ePrint Archive (March 2008)
[46] Simon, D.R.: Findings collisions on a one-way street: Can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
[47] Wee, H.M.: One-way permutations, interactive hashing and statistically hiding commitments. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 419–433. Springer, Heidelberg (2007)

Breaking RSA Generically Is Equivalent to Factoring

Divesh Aggarwal and Ueli Maurer

Department of Computer Science
ETH Zurich
CH-8092 Zurich, Switzerland
{divesha,maurer}@inf.ethz.ch

Abstract. We show that a generic ring algorithm for breaking RSA in $\mathbb{Z}_N$ can be converted into an algorithm for factoring the corresponding RSA-modulus N. Our results imply that any attempt at breaking RSA without factoring N will be non-generic and hence will have to manipulate the particular bit-representation of the input in $\mathbb{Z}_N$. This provides new evidence that breaking RSA may be equivalent to factoring the modulus.

1 Introduction

Probably the two most fundamental reduction problems in number-theoretic cryptography are to prove or disprove that breaking the RSA system [22] is as hard as factoring integers and that breaking the Diffie-Hellman protocol [9] is as hard as computing discrete logarithms. While the second problem has been solved to a large extent [15,18,2], not much is known about the first for general models of computation. In this paper, we show the equivalence of RSA and factoring for the most general generic model of computation.

1.1 Breaking RSA vs. Factoring

The security of the well-known RSA public-key encryption and signature scheme [22] relies on the assumption that computing e^{th} roots modulo n, which is a product of two primes, is hard. In order to formally state this assumption, we define a pair of random variables, N and E, that are chosen according to a certain joint probability distribution as follows: N is a product of two primes, for example an element chosen uniformly at random from the set of products of two k-bit primes satisfying certain conditions (e.g. [16]), and E is a positive integer[1] such that $\gcd(E, \phi(N)) = 1$. Note that the marginal distribution of N over products of two primes is defined by the joint distribution of (N, E). Throughout this paper, we will assume that the sum of the lengths of N and E is bounded by the security parameter κ and the terms negligible, non-negligible, and polynomial-time are with respect to κ. We state three assumptions below:

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

[1] In principle E can be much larger than N.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 36–53, 2009.

Factoring Assumption: There exists no probabilistic polynomial-time algorithm that, given N, finds a non-trivial factor of N with non-negligible[2] probability.

RSA Assumption: There exists no probabilistic polynomial-time algorithm that, given the pair (N, E) and an element a chosen uniformly at random from $\mathbb{Z}_N^*$, computes $x \in \mathbb{Z}_N^*$ such that $x^E \equiv a$ modulo N with non-negligible probability.

Generic RSA Assumption: There exists no probabilistic polynomial-time *generic* ring algorithm (a class of algorithms that we will introduce later) that, given the pair (N, E) and an element a chosen uniformly at random from $\mathbb{Z}_N^*$, computes $x \in \mathbb{Z}_N^*$ such that $x^E \equiv a$ modulo N with non-negligible probability.

It is easy to see that if the RSA assumption holds then the factoring assumption holds. However it is a long-standing open problem whether the converse is true. Since no progress has been made for general models of computation, it is interesting to investigate reasonable restricted models of computation and prove that in such a model factoring is equivalent to the RSA problem. In a restricted model one assumes that only certain kinds of operations are allowed. Shoup [23], based on the work of Nechaev [20], introduced the concept of generic algorithms which are algorithms that do not exploit any property of the representation of the elements. They proved lower bounds on the complexity of computing discrete logarithms in cyclic groups in the context of generic algorithms. Maurer [17] provided a simpler and more general model for analyzing representation-independent algorithms.

1.2 The Generic Model of Computation

We give a brief description of the model of [17]. The model is characterized by a black-box **B** which can store values from a certain set T in internal state variables $V_0, V_1, V_2, \cdots$. The initial state (the input of the problem to be solved) consists of the values of $[V_0, \ldots, V_\ell]$ for some positive integer ℓ, which are set according to some probability distribution (e.g. the uniform distribution).

The black box **B** allows two types of operations:

- *Computation operations.* For a set Π of operations of some arities on T, a computation operation consists of selecting $f \in \Pi$ (say t-ary) as well as the indices $i_1, \ldots, i_{t+1}$ of $t+1$ state variables. **B** computes $f(V_{i_1}, \ldots, V_{i_t})$ and stores the result in $V_{i_{t+1}}$.
- *Relation Queries.* For a set Σ of relations (of some arities) on T, a query consists of selecting a relation $\rho \in \Sigma$ (say t-ary) as well as the indices $i_1, \ldots, i_t$ of t state variables. The query is replied by the binary output $\rho(V_{i_1}, \ldots, V_{i_t})$ that takes the value 1 if the relation is satisfied, and 0 otherwise.

[2] A function $f(\kappa)$ is considered a non-negligible function in κ if there exists $c > 0$ and $k_0 \in \mathbb{N}$ such that for all $\kappa > k_0$, $|f(\kappa)| > \frac{1}{\kappa^c}$.

For this paper, we only consider the case $t = 2$ and the only relation queries we consider are equality queries.

An algorithm in this model is characterized by its interactions with the black box $\mathbf{B}$. The algorithm inputs operations (computation operations and relation queries) to the black box, and the replies to the relation queries are input to the algorithm. The complexity of an algorithm for solving any problem can be measured by the number of operations it performs on $\mathbf{B}$.

For this paper, the set T is $\mathbb{Z}_N$. Also $\ell = 1$ and V_0 is always set to be the unit element 1 of $\mathbb{Z}_N$ and V_1 is the value a whose E^{th} root is to be computed. A *generic ring algorithm (GRA)* is an algorithm that is just allowed to perform the ring operations, i.e., addition and multiplication as well as the inverse ring operations (subtraction and division), and to test for equality (i.e., make an equality query). In this model, for example, GRAs on $\mathbb{Z}_N$ correspond to $\Pi = \{+, -, \cdot, /\}$ and $\Sigma = \{=\}$. A *straight-line program (SLP)* on $\mathbb{Z}_N$, which is a deterministic algorithm that is just allowed to perform ring operations, corresponds to the case where Σ is the empty set, i.e., no equality tests are possible.

Many results in the literature are restricted in that they exclude the inverse operations, but since these operations are easy[3] to perform in $\mathbb{Z}_N$, they should be included as otherwise the results are of relatively limited interest. Note that division by non-invertible elements of $\mathbb{Z}_N$ is not defined. This can be modeled in the above generic model by having the black-box $\mathbf{B}$ send an "exception" bit b to the algorithm and leaving the corresponding state variable undefined whenever there is a division by a non-invertible element. In order to avoid having to handle these exceptions, we instead describe a black-box $\widetilde{\mathbf{B}}$ that (we will show) is almost as powerful as $\mathbf{B}$, but easier to work with.

$\widetilde{\mathbf{B}}$ stores values from $\mathbb{Z}_N \times \mathbb{Z}_N$. V_0 and V_1 are set to be $(1, 1)$ and $(x, 1)$ respectively. The operations in $\{+, -, \cdot, /\}$ are defined on $\mathbb{Z}_N \times \mathbb{Z}_N$ as follows (for $\alpha, \beta, \gamma, \delta \in \mathbb{Z}_N$):

$$(\alpha, \beta) \circ (\gamma, \delta) = \begin{cases} (\alpha\delta + \beta\gamma, \beta\delta) & \text{if } \circ \text{ is } + \\ (\alpha\delta - \beta\gamma, \beta\delta) & \text{if } \circ \text{ is } - \\ (\alpha\gamma, \beta\delta) & \text{if } \circ \text{ is } \cdot \\ (\alpha\delta, \beta\gamma) & \text{if } \circ \text{ is } / \end{cases}$$

If (α, β) and (γ, δ) are queried for equality, 1 is returned if $\alpha\delta = \beta\gamma$, and 0 otherwise.

The interpretation of $\widetilde{\mathbf{B}}$ is that if an internal state variable in $\widetilde{\mathbf{B}}$ takes a value (α, β) then, on inputting the same sequence of operations to $\mathbf{B}$, the corresponding internal state variable in $\mathbf{B}$ takes the value α/β if there was no "exception". We will show in Section 2.2 that any GRA (for solving a certain computation problem) interacting with $\mathbf{B}$ can be converted into a GRA interacting with $\widetilde{\mathbf{B}}$ such that both have essentially the same success probability and complexity of the same order.

[3] Division of an element a by b for $a, b \in \mathbb{Z}_N$ can be performed easily by first computing b^{-1} using Euclid's algorithm and then computing $a \cdot b^{-1}$ in $\mathbb{Z}_N$.

1.3 Related Work and Contributions of This Paper

Research on the relation between RSA and factoring comes in two flavours. There have been results giving evidence against (e.g. [3,12]) and in favour of (e.g. [4,13]) the assumption that breaking RSA is equivalent to factoring.

Boneh and Venkatesan [3] showed that any SLP that factors N by making at most a logarithmic number of queries to an oracle solving the Low-Exponent RSA (LE-RSA) problem (the RSA problem when the public exponent E is small) can be converted into a real polynomial-time algorithm for factoring N. This means that if factoring is hard, then there exists no straight-line reduction from factoring to LE-RSA that makes a small number of queries to the LE-RSA oracle. Joux et al [12] showed that computing e-th roots modulo N is easier than factoring N with currently known methods, given sub-exponential access to an oracle outputting the roots of numbers of the form $x_i + c$.

Brown [4] showed that if factoring is hard then the LE-RSA problem is intractable for SLPs with $\Pi = \{+, -, \cdot\}$. More precisely, he proved that an efficient SLP for breaking LE-RSA can be transformed into an efficient factoring algorithm. Leander and Rupp [13] generalized the result of [4] to GRAs which, as explained above, can test the equality of elements. Again, division is excluded ($\Pi = \{+, -, \cdot\}$).

Another theoretical result about the hardness of the RSA problem is due to Damgård and Koprowski [8]. They studied the problem of root extraction in finite groups of unknown order and proved that the RSA problem is intractable with respect to generic group algorithms. This corresponds to excluding addition, subtraction and division from the set of operations ($\Pi = \{\cdot\}$).

Our results generalize the previous results in several ways. (Actually, Theorem 1 appears to be the most general statement about the equivalence of factoring and breaking RSA in a generic model.)

- First, compared to [4,13] we consider the full-fledged RSA problem (not only LE-RSA) with exponent E of arbitrary size, even with bit-size much larger than that of N.
- Second, we allow for *randomized* GRAs, an extension less trivial than it might appear.
- Third, compared to [8,4,13] we consider the unrestricted set of ring operations, including division. This generalization is important since there are problems that are easy to solve in our generic ring model but are provably hard to solve using the model without division[4].
  Actually, as has been pointed out in [4], computing the multiplicative inverse of a random element in $\mathbb{Z}_N$ generically is hard if $\Pi = \{+, -, \cdot\}$.
- Fourth, compared to [13] we give an explicit algorithm that factors N given a GRA computing E^{th} roots. In [13], the reduction from GRAs to SLPs is only non-uniform.

[4] In [4], the author has mentioned and given justification for the fact that most results of his paper will extend to SLPs with division.

The problem we solve has been stated as an open problem in [8] and [4].

The rest of the paper is structured as follows: In Section 2, we introduce basic definitions and notations and show a few preliminary results. In Section 3, we prove our main result. Section 4 provides some conclusions and lists some open problems.

2 Preliminaries

2.1 Straight-Line Programs

Straight-line programs for a ring are deterministic algorithms that perform a sequence of ring operations. Thus an SLP corresponds to $\Pi = \{+, -, \cdot, /\}$ and $\Sigma = \{\}$ in the model of [17]. More concretely:

Definition 1. An L-step *straight-line program (SLP)* S is a sequence of triples $(i_2, j_2, \circ_2), \ldots, (i_L, j_L, \circ_L)$ such that $0 \leq i_k, j_k < k$ and $\circ_k \in \{+, -, \cdot, /\}$.

The interpretation for this is that if x is considered a variable, then the SLP S computes a sequence of rational functions $f_2, \ldots, f_L (= f^S)$, where, $f_0 = 1$, $f_1 = x$ and $f_k = f_{i_k} \circ_k f_{j_k}$ for $2 \leq k \leq L$. Thus, an SLP can be interpreted as the evaluation of a rational function f^S in x given by a pair of polynomials (P^S, Q^S) representing the numerator and the denominator of f^S, respectively, as follows.

1. Let $P_0^S = 1, Q_0^S = 1, P_1^S = x, Q_1^S = 1$.
2. For $2 \leq k \leq L$, $(P_k^S, Q_k^S) = (P_{i_k}^S, Q_{i_k}^S) \circ_k (P_{j_k}^S, Q_{j_k}^S)$ is given by:

$$(P_k^S, Q_k^S) = \begin{cases} (P_{i_k}^S \cdot Q_{j_k}^S + P_{j_k}^S \cdot Q_{i_k}^S, Q_{i_k}^S \cdot Q_{j_k}^S) & \text{if } \circ_k \text{ is } + \\ (P_{i_k}^S \cdot Q_{j_k}^S - P_{j_k}^S \cdot Q_{i_k}^S, Q_{i_k}^S \cdot Q_{j_k}^S) & \text{if } \circ_k \text{ is } - \\ (P_{i_k}^S \cdot P_{j_k}^S, Q_{i_k}^S \cdot Q_{j_k}^S) & \text{if } \circ_k \text{ is } \cdot \\ (P_{i_k}^S \cdot Q_{j_k}^S, Q_{i_k}^S \cdot P_{j_k}^S) & \text{if } \circ_k \text{ is } / \end{cases}$$

3. Output $(P^S, Q^S) = (P_L^S, Q_L^S)$.

Therefore, from now on, we identify the SLP S with the pair of polynomials (P^S, Q^S).

Lemma 1. *P_k^S and Q_k^S are polynomials of degree at most 2^k.*

Proof. The proof is by induction on k. The claim is trivially true for $k = 0$. We assume that P_r^S and Q_r^S are polynomials of degree at most 2^r for all $r < k$. So, since $i_k, j_k \leq k - 1$, $deg(P_{i_k}^S)$, $deg(Q_{i_k}^S)$, $deg(P_{j_k}^S)$ and $deg(Q_{j_k}^S)$ are all at most 2^{k-1}. This implies, by the definition of (P_k^S, Q_k^S), that the degree of P_k^S and Q_k^S is at most $2^{k-1} + 2^{k-1} = 2^k$. □

Next, we show that any SLP can be "converted" into an SLP that does not require the division operation, i.e., for which $\Pi = \{+, -, \cdot\}$, without increasing the complexity by more than a constant factor. A result similar to the following but with a factor of 6 instead of 4 has been proven independently in [11].

Lemma 2. *For any L-step SLP S with $\Pi = \{+, -, \cdot, /\}$ that computes the rational function f^S given by (P^S, Q^S), there exists a $4L$-step SLP with $\Pi = \{+, -, \cdot\}$ that computes P^S and Q^S.*

Proof. We prove this by induction on L. The result is trivial for $L = 0$. We suppose it is true for $L = L'$. Therefore there exists an SLP S' of length $\ell \leq 4L'$ that uses only the operations $\{+, -, \cdot\}$ and computes P_k^S and Q_k^S for $1 \leq k \leq L'$. Let this program compute the polynomials R_i for $1 \leq i \leq \ell$. Let $L = L' + 1$. Now consider the following cases:

- Case (i): $\circ_L$ is $+$.
  Let $R_{\ell+1} = P_{i_L}^S \cdot Q_{j_L}^S$, $R_{\ell+2} = P_{j_L}^S \cdot Q_{i_L}^S$, $R_{\ell+3} = R_{\ell+1} + R_{\ell+2} = P_L^S$ and $R_{\ell+4} = Q_{i_L}^S \cdot Q_{j_L}^S = Q_L^S$.
- Case (ii): $\circ_L$ is $-$.
  Let $R_{\ell+1} = P_{i_L}^S \cdot Q_{j_L}^S$, $R_{\ell+2} = P_{j_L}^S \cdot Q_{i_L}^S$, $R_{\ell+3} = R_{\ell+1} - R_{\ell+2} = P_L^S$ and $R_{\ell+4} = Q_{i_L}^S \cdot Q_{j_L}^S = Q_L^S$.
- Case (iii): $\circ_L$ is $\cdot$.
  Let $R_\ell = P_{i_L}^S \cdot P_{j_L}^S = P_L^S$ and $R_{\ell+2} = Q_{i_L}^S \cdot Q_{j_L}^S = Q_L^S$.
- Case (iv): $\circ_L$ is $/$.
  Let $R_{\ell+1} = P_{i_L}^S \cdot Q_{j_L}^S = P_L^S$ and $R_{\ell+2} = Q_{i_L}^S \cdot P_{j_L}^S = Q_L^S$.

By induction hypothesis, SLP S' computes P_k^S and Q_k^S for $1 \leq k \leq L'$. We can extract from S', the indices corresponding to each of P_k^S and Q_k^S for $1 \leq k \leq L'$. Therefore, in each of the cases mentioned above, we get an SLP of length at most $\ell + 4 \leq 4(L' + 1) = 4L$ that computes P_k^S and Q_k^S for $1 \leq k \leq L' + 1$. □

Note that the SLP described in the above lemma does not compute the rational function f^S but only computes the two polynomials P^S and Q^S. This, however, is sufficient for our purpose.

2.2 Generic Ring Algorithms

A deterministic generic ring algorithm can be seen as a generalized SLP that also allows equality queries. More concretely:

Definition 2. An L-step *deterministic generic ring algorithm (deterministic GRA)* G is an algorithm that, in the k^{th} step, for $2 \leq k \leq L$, outputs an operation of the form $(i_k, j_k, \circ_k)$, where $0 \leq i_k, j_k < k$ are some positive integers and $\circ_k \in \{+, -, \cdot, /, eq\}$. It takes as input an "exception" bit b which is 1 if the operation is not defined, and 0 otherwise. If b is 0 and $\circ_k$ is eq, it takes another input bit (as the result of the equality query).

The interpretation for this definition is that if x is considered a variable, then G computes a sequence of rational functions $f_2, \ldots, f_L$, where $f_0 = 1$, $f_1 = x$, and $f_k = f_{i_k} \circ_k f_{j_k}$ if $\circ_k \in \{+, -, \cdot, /\}$ and $f_k = f_{k-1}$ if $\circ_k$ is eq for $2 \leq k \leq L$. Also, $f_k = \perp$ if f_{j_k} is not invertible and $\circ_k$ is $/$ or one of f_{i_k} and f_{j_k} is $\perp$.

If G is to be executed on an input $a \in \mathbb{Z}_N$, this is modeled, as mentioned in Section 1.2, by G interacting with the black-box $\mathbf{B}$ where an operation of the

form (i, j, eq) is an equality query on the i^{th} and j^{th} elements in $\mathbf{B}$. We now show that if a GRA G is successful in solving a certain computation problem for $\mathbf{B}$ with probability γ, then the sum of the probabilities that there is a non-trivial non-invertible element computed by G in $\mathbf{B}$ (in which case this element can be used to factorize N) and that there is a GRA (of double the length of G) that is successful in solving the same computational problem for $\widetilde{\mathbf{B}}$, where $\widetilde{\mathbf{B}}$ is as defined in Section 1.2, is at least γ. Since our aim is to reduce the problem of factoring N to computing the E-th roots modulo N in the generic model of Section 1.2, it is therefore sufficient to prove the result in the generic model replacing $\mathbf{B}$ by $\widetilde{\mathbf{B}}$.

Consider a black box $\mathbf{B}'$ that behaves exactly like $\mathbf{B}$ except that it does not output an exception bit. Any GRA G interacting with $\mathbf{B}$ can be converted into a GRA G' interacting with $\mathbf{B}'$ as follows, such that $\mathbf{B}$ and $\mathbf{B}'$ have identical behaviour internally. For each new value computed inside $\mathbf{B}'$, G' makes an equality query of the computed value and 0 and stores the result internally. Also, it maintains internally a list of indices of state variables which are undefined. Then, G' can internally simulate the exception bit that G receives from $\mathbf{B}$ by setting the exception bit as 1 if and only if there is a division by 0 in $\mathbf{B}'$ or an operation is performed on two values, one of which is undefined. Thus G' performs at most twice the number of steps as G and gets the same input as G if there is no division by a non-trivial non-invertible element in $\mathbf{B}$.

By definition of the operations $\{+, -, \cdot, /, eq\}$ in $\widetilde{\mathbf{B}}$, if a GRA performing a sequence of operations computes a value α in $\mathbf{B}'$ and the same sequence of operations computes a pair (α_1, α_2) in $\widetilde{\mathbf{B}}$, then $\alpha = \frac{\alpha_1}{\alpha_2}$ if α is not undefined.

Thus any GRA G that computes a certain value in $\mathbf{B}$ (without any division by a non-trivial non-invertible element of $\mathbb{Z}_N$) can be converted into a GRA G' that computes a pair of values such that if the first element of this pair is divided by the second, we get the value computed in $\mathbf{B}$ (unless, this value is undefined). Hence, from now on, we will consider only the black-box $\widetilde{\mathbf{B}}$ and therefore assume that the corresponding GRA is never returned 1 as the "exception" bit. As a result, we can ignore the "exception" bit.

Note that for a given sequence of bits input to it, a deterministic GRA G behaves like an SLP (that performs a trivial operation of copying the previous value whenever $\circ_k$ is eq). A deterministic GRA can be interpreted as a binary tree T^G with each vertex corresponding to an operation from the set $\{+, -, \cdot, /, eq\}$. The vertices corresponding to an eq operation have two children and all other vertices have one child. The edges between a vertex corresponding to an eq operation and its left and right child are labeled 0 and 1, respectively, while all the other edges do not have a label. An execution of G corresponds to a path from the root to a leaf of T^G. The choice of the subsequent vertex at each vertex corresponding to an operation eq is made by choosing the edge that has as label the bit input to G at the corresponding step. The sequence of operations corresponding to vertices along each path from the root to a vertex of T^G is an SLP, and so we can associate a pair of polynomials (using the definition of

SLP) with each vertex of T^G. The pair of polynomials associated with vertex v is denoted by (P^v, Q^v).

Definition 3. A *randomized generic ring algorithm* $\mathcal{G}$ is a GRA where the choice of the operation at each step is randomized. A randomized GRA can be understood as a random variable whose values are deterministic GRAs.

2.3 Mathematical Preliminaries

In this section we introduce some notations and prove some results about the mathematical structures used in this paper.

For integers a, b, c, we denote by $a \equiv_c b$, that a is congruent to b modulo c. For any event E, we denote the probability of E by $\mathsf{P}(\mathsf{E})$.

For the rest of the paper, we assume that the random variable N takes values from some set $\mathcal{N}$. Furthermore, (n, e), where $n = pq$, denotes a value taken by the pair (N, E). By $\mathbb{Z}_n[x]$, we denote the ring of polynomials in x with coefficients in $\mathbb{Z}_n$ and for $h(x) \in \mathbb{Z}_n[x]$ by $\mathbb{Z}_n[x]/h(x)$ quotient of the ring $\mathbb{Z}_n[x]$ by a principal ideal generated by an irreducible polynomial $h(x)$. For $P(x) \in \mathbb{Z}_n[x]$, we define the following.

- Let $\nu_n(P)$ be the fraction of roots of P in $\mathbb{Z}_n$, i.e.,

$$\nu_n(P) = \frac{|\{x \in \mathbb{Z}_n | P(x) \equiv_n 0\}|}{n} .$$

  Similarly we define $\nu_p(P)$ and $\nu_q(P)$.
- The fraction of elements a in $\mathbb{Z}_n$ such that $P(a)$ has a non-trivial greatest common divisor with n is defined as $\eta_n(P)$,

$$\eta_n(P) = \frac{|\{x \in \mathbb{Z}_n | \gcd(P(a), n) \notin \{1, n\}\}|}{n} .$$

We prove three lemmas that we will need later. The reader may skip to Section 3 and return to the following lemmas when they are referenced.

Lemma 3. *For any $P(x) \in \mathbb{Z}_n[x]$, if $\nu_n(P) \in [\delta, 1-\delta]$, then $\eta_n(P) \geq \delta^{\frac{3}{2}}$.*

Proof. We denote $\nu_p(P)$ and $\nu_q(P)$ by ν_p and ν_q, respectively. By the Chinese remainder theorem, $\nu_n(P) = \nu_p \cdot \nu_q$ and $\eta_n(P) = \nu_p(1-\nu_q) + \nu_q(1-\nu_p)$. Using $\delta \leq \nu_p \cdot \nu_q \leq 1 - \delta$, we obtain

$$\begin{aligned}
\eta_n(P) &= \nu_p(1-\nu_q) + \nu_q(1-\nu_p) = \nu_p + \nu_q - 2\nu_p \cdot \nu_q \\
&= (\sqrt{\nu_p} - \sqrt{\nu_q})^2 + 2\sqrt{\nu_p \cdot \nu_q} - 2\nu_p \cdot \nu_q \\
&\geq 2\sqrt{\nu_p \cdot \nu_q} - 2\nu_p \cdot \nu_q = 2\sqrt{\nu_p \cdot \nu_q}(1 - \sqrt{\nu_p \cdot \nu_q}) \\
&\geq 2\sqrt{\delta}(1 - \sqrt{1-\delta}) \\
&\geq 2\sqrt{\delta}(1 - (1 - \tfrac{\delta}{2})) = \delta^{\frac{3}{2}} .
\end{aligned}$$

□

Lemma 4. *Let p be a prime. A random monic polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree d is irreducible in $\mathbb{Z}_p[x]$ with probability at least $\frac{1}{2d}$ and has a root in $\mathbb{Z}_p$ with probability at least $1/2$.*

Proof. From the distribution theorem of monic polynomials (see, e.g., [14]) it follows that the number of monic irreducible polynomials of degree d over F_p is at least $\frac{p^d}{2d}$. Therefore $f(x)$ is an irreducible polynomial over $\mathbb{Z}_p$ with probability at least $\frac{1}{2d}$.

The number of monic polynomials over $\mathbb{Z}_p$ with at least one root is:

$$\sum_{l=1}^{d}(-1)^{l-1}\binom{p}{l}p^{d-l} .$$

This can be seen by applying the principle of inclusion and exclusion. The terms in this summation are in decreasing order of their absolute value. So, taking the first two terms, this sum is greater than $\binom{p}{1}p^{d-1} - \binom{p}{2}p^{d-2}$ which is greater than $\frac{p^d}{2}$. Hence the probability that $f(x)$ has a root in $\mathbb{Z}_p$ is at least $1/2$.[5] □

Lemma 5. *For any discrete random variable X that takes values in [0,1] and for any $\tau > 0$, $\mathsf{P}(X \geq \tau) \geq E[X] - \tau$.*

Proof.

$$\begin{aligned} \mathrm{E}[X] &= \sum_{x \geq \tau} x \cdot \mathsf{P}(X = x) + \sum_{x < \tau} x \cdot \mathsf{P}(X = x) \\ &\leq \sum_{x \geq \tau} \mathsf{P}(X = x) + \sum_{x < \tau} \tau \cdot \mathsf{P}(X = x) \\ &\leq \mathsf{P}(X \geq \tau) + \tau \ , \end{aligned}$$

which implies the result. □

3 The Main Theorem

3.1 Statement of the Theorem

As mentioned earlier, in this paper we restrict our attention to the case where the adversary is only allowed to use a GRA to solve the RSA problem. We refer to the RSA assumption in this case as the generic RSA assumption for the random variables (N, E) that was introduced in Section 1.1.

We state the main result of the paper.

[5] Note that, by a careful analysis, it is possible to prove a better lower bound on the probability that $f(x)$ has a root in $\mathbb{Z}_p$ but a lower bound of $1/2$ is sufficient for our purpose.

Theorem 1. *For any pair (N, E), the factoring assumption holds for N implies that the generic RSA assumption holds for (N, E).*

Remark: In the proof of this theorem, we give an algorithm that, for every $n \in \mathcal{N}$ for which there is a GRA that computes the e^{th} root of a uniformly random element chosen from $\mathbb{Z}_n$, factors n with overwhelming probability. The factoring assumption and the generic RSA assumption are, however, stated for the random variable N and not for a fixed n, as the factoring problem would otherwise not be defined reasonably, and also the terms polynomial-time and non-negligible would not make sense for a fixed n. Hence, our proof actually proves a stronger statement than Theorem 1.

3.2 Proof of the Theorem

3.2.1 Overview of the Proof

In Section 3.2.2, we show that an SLP that computes e^{th} roots can be used to factor n. Then, in Section 3.2.3, we show that from a deterministic GRA that computes e^{th} roots we can either obtain an SLP that computes e^{th} roots or a factor of n. In Section 3.2.4, we generalize the results of Section 3.2.3 from deterministic GRAs to randomized GRAs. In Section 3.2.5, we combine the results of Section 3.2.2 and 3.2.4 to show that a randomized GRA that computes e^{th} roots can be used to give an algorithm for factoring n.

3.2.2 The Proof for Straight-Line Programs

In this section we give an algorithm that, with non-negligible probability, factors n given access to an SLP that, with non-negligible probability, computes the e^{th} root of an element chosen uniformly at random from $\mathbb{Z}_n$.

For polynomials $b(x), c(x) \in \mathbb{Z}_n[x]$, let $\gcd_p(b(x), c(x))$ and $\gcd_q(b(x), c(x))$ be the greatest common divisor of the polynomials modulo p and q, respectively. The following proposition is easy to see.

Proposition 1. *Let $b(x), c(x) \in \mathbb{Z}_n[x]$. Then:*

- *If Euclid's algorithm, when run on $b(x)$ and $c(x)$, fails*[6]*, some step of the algorithm yields a non-trivial non-invertible element of $\mathbb{Z}_n$. We denote this element as $H(b(x), c(x))$.*
- *If $deg(gcd_p(b(x), c(x))) \neq deg(gcd_q(b(x), c(x)))$, then Euclid's algorithm, when run on $b(x)$ and $c(x)$, fails.*

Lemma 6. *For all $\epsilon > 0$, $\mu > 0$, and $L \in \mathbb{N}$, there exists an algorithm of time complexity $O(L^3 + log^3(e))$ that, for every SLP S such that $\nu_n((P^S)^e - x(Q^S)^e) \geq \mu$ and $Q^S(x)$ is not the zero polynomial, returns a factor of n with probability $\frac{\mu}{8(L+log(e))}$.*

[6] Euclid's Algorithm could fail since $\mathbb{Z}_n[x]$ is not a Euclidean domain.

Proof. Let $f(x) = P^S(x)^e - x \cdot Q^S(x)^e$. Then $\nu_n(f) \geq \mu$. By Lemma 1, $deg(f) \leq 2^L e + 1$. By Lemma 2, there is a $4L$-step SLP that uses only the operations $\{+, -, \cdot\}$ and generates the polynomials $P^S(x)$ and $Q^S(x)$. Given $P^S(x)$ and $Q^S(x)$, $P^S(x)^e$ and $Q^S(x)^e$ can be computed in $2\lceil log(e) \rceil$ steps each. Therefore there is an SLP S_1 with at most $4L + 4\lceil log(e) \rceil + 2$ steps that computes $f^{S_1}(x) = f(x)$. For the factoring algorithm, we use this SLP. Let $d = L + \lceil log(e) \rceil$. The factoring algorithm proceeds as follows:

Algorithm 1. Factoring Algorithm

Input: n, SLP S_1
Output: A factor of n

1 Choose a monic polynomial $h(x)$ uniformly at random from all monic polynomials of degree d in $\mathbb{Z}_n[x]$;
2 Compute $h'(x)$, the derivative of $h(x)$ in $\mathbb{Z}_n[x]$;
3 Choose a random element $r(x) \in \mathbb{Z}_n[x]/h(x)$;
4 Compute $z(x) = f(r(x))$ in $\mathbb{Z}_n[x]/h(x)$ using SLP S_1;
5 Run Euclid's algorithm in $\mathbb{Z}_n[x]$ on $h(x)$ and $z(x)$. If this fails return $\gcd(n, H(h(x), z(x)))$;
6 Run Euclid's algorithm in $\mathbb{Z}_n[x]$ on $h(x)$ and $h'(x)$. If this fails return $\gcd(n, H(h(x), h'(x)))$;

By Proposition 1, if Euclid's algorithm fails in step 5 or step 6, then we get a factor of n.

Now we compute the success probability of the algorithm. By Lemma 4, the probability that $h(x)$ is irreducible modulo q and has a root modulo p is at least $\frac{1}{2d} \cdot \frac{1}{2} = \frac{1}{4d}$. We assume that this is the case for the rest of the proof.

Let this root of $h(x)$ modulo p be s. Therefore $(x - s) \mid h(x)$ in $\mathbb{Z}_p[x]$. We analyze two cases.

- CASE 1: $(x - s)^2 \mid h(x)$ in $\mathbb{Z}_p[x]$.
  This implies $(x - s) \mid \gcd_p(h(x), h'(x))$. However, since $h(x)$ is irreducible in $\mathbb{Z}_q[x]$, $\gcd_q(h(x), h'(x))$ has degree 0. Therefore $\gcd_p(h(x), h'(x))$ and $\gcd_q(h(x), h'(x))$ have different degree, which implies, by Proposition 1, that Euclid's algorithm on $h(x)$ and $h'(x)$ fails and hence step 6 yields a factor of n.
- CASE 2: $(x - s)^2 \nmid h(x)$ in $\mathbb{Z}_p[x]$.
  Let $h(x) = h_1(x) \cdot (x - s)$ in $\mathbb{Z}_p[x]$. Then:

  $$\mathbb{Z}_n[x]/h(x) \cong \mathbb{Z}_p[x]/h(x) \times \mathbb{Z}_q[x]/h(x) \cong \mathbb{Z}_p[x]/(x-s) \times \mathbb{Z}_p[x]/h_1(x) \times \mathbb{F}_{q^d} ,$$

  because $\mathbb{Z}_q[x]/h(x) \cong \mathbb{F}_{q^d}$ (the finite field containing q^d elements) as $h(x)$ is irreducible in $\mathbb{Z}_q[x]$ by our assumption.
  Under this isomorphism, let $r(x)$ maps to the triple

  $$(r(s) \bmod p,\ u(x),\ r_q(x)) ,$$

  and $z(x)$ to the triple

$$(z(s) \bmod p,\ v(x),\ z_q(x))\ ,$$

where $r_q(x)$ and $z_q(x)$ are the reductions of $r(x)$ and $z(x)$ modulo q. Since $r(x)$ is uniformly random in $\mathbb{Z}_n[x]/h(x)$, $r(s)$ is uniformly random in $\mathbb{Z}_p[x]/(x-s) \cong \mathbb{Z}_p$. This implies

$$\mathsf{P}\big(z(s) \equiv_p 0\big) \ = \ \mathsf{P}\big(f(r(s)) \ \equiv_p \ 0\big) \ \geq \ \mathsf{P}\big(f(r(s)) \equiv_n 0\big) \ \geq \ \mu\ .$$

Therefore, with probability at least μ, $(x-s)$ divides $z(x)$ in $\mathbb{Z}_p[x]$, which implies $\mathsf{P}((x-s) \mid \gcd_p(z(x), h(x))) \geq \mu$. Since $r(x)$ is uniformly random in $\mathbb{Z}_n[x]/h(x)$, $r_q(x)$ is uniformly random in $\mathbb{Z}_q[x]/h(x) \cong \mathbb{F}_{q^d}$. A polynomial over a finite field can have at most as many roots as the degree of the polynomial. Therefore, for random x,

$$\mathsf{P}\big(z_q(x) = 0\big) \ = \ \mathsf{P}\big(f(r_q(x)) = 0\big) \ \leq \ \frac{deg(f)}{q^d} \ \leq \ \frac{2^L e + 1}{q^d} \ \leq \ \frac{1}{2}\ ,$$

using the fact that $d = L + \lceil log(e) \rceil$ and $deg(f) \leq 2^L e + 1$. Hence, $\mathsf{P}(z_q(x) \neq 0) \geq \frac{1}{2}$. The condition $z_q(x) \neq 0$ implies that $\gcd_q(z(x), h(x))$ has degree 0 because $h(x)$ is irreducible modulo q.

Therefore the probability that Euclid's algorithm run on $h(x)$ and $z(x)$ fails is at least $\frac{1}{4d} \cdot \mu \cdot \frac{1}{2} = \frac{\mu}{8d}$.

Now we compute the time complexity of one run of the loop. Generating random $h(x)$ and $r(x)$ and computing the derivative requires $O(d)$ operations in $\mathbb{Z}_n$. Each operation in $\mathbb{Z}_n[x]/h(x)$ can be implemented by at most d^2 operations in $\mathbb{Z}_n$. The function f is computed in $\mathbb{Z}_n$ using an at most $(4L + 4\lceil log(e) \rceil + 2)$-step SLP S_1. Therefore, $f(r(x)) = z(x)$ can be computed in time $O(d^2 \cdot L + d^2 \cdot log(e)) = O(d^3)$. Euclid's algorithm on $z(x)$ and $h(x)$ and on $h(x)$ and $h'(x)$ can be performed by $O(d^2)$ operations. Thus, the running time of the algorithm is $O(d^3)$. □

3.2.3 From Deterministic GRAs to SLPs

In this section we give an algorithm that, given access to a deterministic GRA that computes e^{th} roots in $\mathbb{Z}_n$, outputs either a factor of n or an SLP that computes e^{th} roots in $\mathbb{Z}_n$.

Definition 4. For a deterministic GRA G, let $\lambda_n(G)$ denote the probability that G, when run on an input a chosen uniformly at random from $\mathbb{Z}_n$, is successful in computing the e^{th} root of a.

Lemma 7. *For all $\epsilon > 0$ and $L \in \mathbb{N}$, there exists an algorithm (Algorithm 2) of time complexity $O((\frac{L}{\epsilon})^{5/2}$ that, given access to an L-step deterministic GRA G, with probability $1 - \epsilon$, either outputs a factor of n or an L-step SLP S such that $\nu_n((P^S)^e - x(Q^S)^e) \geq \lambda_n(G) - \frac{\epsilon}{2}$.*

Proof. Consider the tree T^G(see Section 2.2). With each vertex v of T^G, we can associate an SLP, and hence a sequence $(P_1^v, Q_1^v), \ldots, (P_k^v, Q_k^v)$ given by the sequence of operations along the path from the root of T^G to that vertex.

Let $\delta = \frac{\epsilon}{2L}$. We classify the vertices corresponding to equality queries in T^G into two kinds of vertices-*extreme* and *non-extreme* vertices. For a vertex v corresponding to an equality query (i_k, j_k, eq), if

$$\nu_n(P_{i_k}^v \cdot Q_{j_k}^v - P_{j_k}^v \cdot Q_{i_k}^v) \in [\delta, 1-\delta] \,,$$

then we call v a *non-extreme* vertex and otherwise we call v an *extreme* vertex.

Let T_{ex}^G be the tree obtained from T^G by truncating the sub-tree rooted at v for all non-extreme vertices v. Therefore all non-extreme vertices present in T_{ex}^G are at the leaves. Also, we can assume, without loss of generality, that the last step of G is not an equality query since that would be of no use. Hence there cannot be an extreme vertex at a leaf vertex of T_{ex}^G.

Let $v^\star(T_{ex}^G)$ be the unique leaf vertex of T_{ex}^G reached by starting from the root and inputting, for all extreme vertices v, the bit 1 to G if $\nu_n(P_{i_k}^v \cdot Q_{j_k}^v - P_{j_k}^v \cdot Q_{i_k}^v) \in [0, \delta)$ and the bit 0 if $\nu_n(P_{i_k}^v \cdot Q_{j_k}^v - P_{j_k}^v \cdot Q_{i_k}^v) \in (1-\delta, 1]$. Note that $v^\star(T_{ex}^G)$ is either a non-extreme vertex or corresponds to a computation operation. We call the path in T_{ex}^G from the root to the vertex $v^\star(T_{ex}^G)$ the *dominating path* because this is the path that is most likely to be taken if G is run on a random input from $\mathbb{Z}_n$ as we make the most likely choice at each equality test (assuming δ to be small).

Let $M = \lceil \frac{L^{3/2}}{(\epsilon/2)^{5/2}} \rceil$. Consider algorithm 2, as given on the next page.

The intuition is that when executing the GRA for a random element a, either all the equality test one encounters are irrelevant in the sense that the probability that the outcome depends on a is very small, and hence the execution corresponds to an SLP, or the relevant equality test encountered during the execution can be used to factor.

At each equality query, it tries to find a factor of n using the two pairs of polynomials that are compared. If it fails, it outputs the SLP S as the sequence of computation operations along one of the paths from the root to a leaf of T_G. This path corresponds to a unique path in T_{ex}^G. This path is chosen by generating, for each equality query, a uniformly random element in $\mathbb{Z}_n$ and then testing the equality of the two polynomials on this element, and choosing the subsequent vertex based on the result of this equality test. Algorithm 2 is successful with high probability (as shown below) if this path is the dominating path, i.e., if it reaches the vertex $v^\star(T_{ex}^G)$.

Line 7 of the algorithm is a trivial operation and could be avoided but is there so that we do not have to manipulate the indices of the straight line program output by the algorithm.

The SLP S output by Algorithm 2 corresponds to one of the paths from the root to a leaf of T^G which defines a unique path from the root to a leaf of T_{ex}^G. Let the leaf vertex of T_{ex}^G in which this path terminates be v_S. Note that v_S might not be a leaf vertex of T_G.

Algorithm 2.

Input: GRA G, n
Output: A factor of n or an SLP S

1 Initialize S to be the empty sequence;
2 **for** $k \leftarrow 2$ **to** L **do**
3 Get the operation $\{i_k, j_k, \circ_k\}$ from G;
4 **if** $\circ_k \in \{+, -, \cdot, /\}$ **then**
5 Append $\{i_k, j_k, \circ_k\}$ to S;
6 **else** /* Here, $\circ_k$ is eq */
7 Append $\{k-1, 0, \cdot\}$ to S;
8 **for** $i \leftarrow 1$ **to** M **do**
9 Generate a random element $x \in_R \mathbb{Z}_n$;
10 Compute $g = \gcd\left(P^S_{i_k}(x) \cdot Q^S_{j_k}(x) - P^S_{j_k}(x) \cdot Q^S_{i_k}(x), n\right)$;
11 **if** $g \notin \{1, n\}$ **then** return g;
12 **end**
13 Generate a random element $x' \in_R \mathbb{Z}_n$;
14 **if** $P^S_{i_k}(x') \cdot Q^S_{j_k}(x') - P^S_{j_k}(x') \cdot Q^S_{i_k}(x') = 0$ **then** return the bit 0 to G **else** return the bit 1 to G;
15 **end**
16 **end**
17 Return S;

If Algorithm 2 outputs S, then let (P^S, Q^S) denote the pair of polynomials corresponding to S.

Let E be the event that $v_S = v^\star(T^G_{ex})$, i.e., that the dominating path is found by Algorithm 2. The event E does not occur if there exists an extreme vertex v in the path from the root of T^G_{ex} to v_S corresponding to (i_k, j_k, eq) such that Algorithm 2 inputs 0 to G and $\nu_n(P^v_{i_k} \cdot Q^v_{j_k} - P^v_{j_k} \cdot Q^v_{i_k}) \in [0, \delta)$ or Algorithm 2 inputs 1 to G and $\nu_n(P^v_{i_k} \cdot Q^v_{j_k} - P^v_{j_k} \cdot Q^v_{i_k}) \in (1-\delta, 1]$. Note that this can happen with probability at most δ at each extreme vertex v and there can be at most L such extreme vertices in the path from the root of T^G_{ex} to v_S. Therefore,

$$\mathsf{P}(\mathsf{E}) = 1 - \mathsf{P}(\overline{\mathsf{E}}) \geq 1 - \delta \cdot L = 1 - \tfrac{\epsilon}{2} .$$

Now we compute the success probability of the algorithm. There are two possible cases depending on whether $v^\star(T^G_{ex})$ is a non-extreme vertex or corresponds to a computation operation.

– CASE 1: $v^\star(T^G_{ex})$ is a non-extreme vertex.
  In this case we show that the factoring algorithm is successful with probability at least $1 - \epsilon$.
  Let F be the event that Algorithm 2 returns a factor of n. We compute $\mathsf{P}(\mathsf{F}|\mathsf{E})$. If E holds, then v_S is a non-extreme vertex. Therefore, by Lemma 3, a factor of n is returned in one test in Step 11 at the equality query corresponding

to v_S with probability at least $\delta^{3/2}$. The total number of times step 11 is repeated for this equality query is M. Therefore[7],

$$\mathsf{P}(\mathsf{F}|\mathsf{E}) \geq 1-(1-\delta^{3/2})^M \geq 1-\exp(-(\delta^{3/2})M) = 1-\exp(-\tfrac{2}{\epsilon}) \geq 1-\tfrac{\epsilon}{2}\ .$$

This implies

$$\mathsf{P}(\mathsf{F}) \geq \mathsf{P}(\mathsf{F}|\mathsf{E})\cdot\mathsf{P}(\mathsf{E}) \geq (1-\tfrac{\epsilon}{2})^2 \geq 1-\epsilon\ .$$

– CASE 2: $v^\star(T_{ex}^G)$ corresponds to a computation operation.
In this case, we show that if the factoring algorithm is not successful, then, with probability $1-\frac{\epsilon}{2}$, we have $\nu_n((P^S)^e - x(Q^S)^e) \geq \lambda_n(G) - \frac{\epsilon}{2}$.
The fraction of inputs $a \in \mathbb{Z}_n$ such that when G is run on a, the corresponding path taken on T_{ex}^G does not terminate in $v^\star(T_{ex}^G)$ is at most $\delta \cdot L = \frac{\epsilon}{2}$ (because the number of extreme vertices in any path from root to a leaf is at most L). This implies,

$$\mathsf{P}_{a\in_R\mathbb{Z}_n}\left(P^{v^\star(T_{ex}^G)}(a)^e - a\cdot Q^{v^\star(T_{ex}^G)}(a)^e \equiv_n 0\right) \ \geq\ \lambda_n(G)-\tfrac{\epsilon}{2}\ .$$

Therefore, if E occurs, then

$$\begin{aligned}\nu_n((P^S)^e - x(Q^S)^e) &= \mathsf{P}_{a\in_R\mathbb{Z}_n}\left(P^{v^\star(T_{ex}^G)}(a)^e - a\cdot Q^{v^\star(T_{ex}^G)}(a)^e \equiv_n 0\right)\\ &\geq \lambda_n(G)-\tfrac{\epsilon}{2}\ .\end{aligned}$$

Hence,

$$\mathsf{P}\left(\nu_n((P^S)^e - x(Q^S)^e) \geq \lambda_n(G)-\tfrac{\epsilon}{2}\right) \ \geq\ \mathsf{P}(\mathsf{E}) \ \geq\ 1-\tfrac{\epsilon}{2}\ .$$

Now, we compute the time complexity of Algorithm 2. The loop in steps 8-12 of the algorithm and steps 5,13 and 14, which are the steps in which computation is being performed, are each executed at most L times. Therefore the time complexity of the algorithm is $O(L\cdot M) = O((\frac{L}{\epsilon})^{5/2})$. □

3.2.4 Handling Randomized GRAs

Here we state a lemma that shows that a result similar to Lemma 7 also holds for *randomized* GRAs. Recall that a randomized GRA $\mathcal{G}$ is understood as a random variable whose values are deterministic GRAs. Let $\mathsf{P}_{\mathcal{G}}$ denote the probability distribution of $\mathcal{G}$. $\lambda_n(\mathcal{G})$ is a random variable. Hence, $\mathrm{E}[\lambda_n(\mathcal{G})]$ is the probability of success of $\mathcal{G}$ in computing the e^{th} root of an element chosen uniformly from $\mathbb{Z}_n$. The proof of Lemma 8 is omitted due to space constraints and can be found in the full version of the paper in the Cryptology ePrint Archive.

Lemma 8. *For all $\epsilon' > 0$, $\mu > 0$, and for every L-step randomized GRA $\mathcal{G}$ such that $E[\lambda_n(\mathcal{G})] \geq \mu$, with probability $\frac{\mu}{2} - \epsilon'$, Algorithm 2 from Lemma 7 either outputs a factor of n or an L-step SLP S such that $\nu_n((P^S)^e - x(Q^S)^e) \geq \frac{\mu}{2}$.*

[7] We use the notation $exp(\cdot)$ to denote exponentiation to the natural base in order to avoid confusion with the public exponent e.

3.2.5 Putting Things Together

In this section, we complete the proof of Theorem 1.

Proof. Suppose there exists a randomized GRA $\mathcal{G}$ that succeeds in breaking RSA with probability μ_n on $\mathbb{Z}_n$ for some $e > 1$. Then, by Lemma 8, with probability $\mu_n - \epsilon'$, Algorithm 2 either returns a factor of n or an SLP S that succeeds in breaking RSA with probability at least $\frac{\mu_n}{2}$, which can be converted into an algorithm that factors n (Algorithm 1) with probability $\frac{\mu_n}{16(L+log(e))}$.

Since the result is true for all $\epsilon' > 0$, let $\epsilon' = \frac{\mu_n}{2}$. Thus, one execution of Algorithm 2 followed by Algorithm 1 (if needed, i.e., if Algorithm 2 does not return a factor of n) runs in time $O(L^3 + log^3(e) + (\frac{L}{\mu_n})^{5/2})$ and returns a factor of n with probability at least $\frac{\mu_n^2}{32(L+log(e))}$. Therefore, the expected number of times the two algorithms need to be repeated in order to get a factor of n is $\frac{32(L+log(e))}{\mu_n^2}$. Hence the expected time complexity of the factoring algorithm is $O((L^3 + log^3(e) + (\frac{L}{\mu_n})^{5/2}) \cdot (\frac{L+log(e)}{\mu_n^2}))$ which is polynomial in L, $log(e)$ and $\frac{1}{\mu_n}$.

If, for a random N, $\mathcal{G}$ succeeds in breaking RSA with probability μ on $\mathbb{Z}_N$ (i.e., if $\mathrm{E}[\mu_N] = \mu$), then by Lemma 5, $\mathsf{P}(\mu_n \geq \frac{\mu}{2}) \geq \mu - \frac{\mu}{2} = \frac{\mu}{2}$. For all n such that $\mu_n \geq \frac{\mu}{2}$, our factoring algorithm runs in time polynomial in L, $log(e)$ and $\frac{1}{\mu}$, which is polynomial if μ is non-negligible. Therefore the factoring algorithm is a probabilistic polynomial time algorithm that succeeds in factoring N with non-negligible probability. □

4 Conclusions and Open Problems

In this paper we showed that if factoring is hard, then no generic ring algorithm can solve the RSA problem efficiently. Also, if there exists an efficient algorithm that can factor N, then we can compute d such that $e \cdot d \equiv_{\phi(N)} 1$, and then the e^{th} root can be computed by computing the d^{th} power, i.e., *generically* in $O(log(d))$ steps. Thus, this proves, in the generic model, the equivalence of factoring and breaking RSA.

It is interesting to note that all arguments in the paper work not just for the RSA equation $x^e = a$ but for any non-trivial polynomial equation in x and a. More concretely, this means the following. We say that a polynomial $M(x, a)$ is trivial if there exists a rational function g such that $M(g(a), a) \equiv 0$ in $\mathbb{Z}[a]$. Then, if there exists a GRA that, given any non-trivial polynomial $M(x, a)$, computes, with non-negligible probability, an x such that $M(x, a) \equiv_N 0$ for a chosen uniformly at random from $\mathbb{Z}_N$, then there exists an algorithm for factoring N.

There are other problems that can be looked at in this model. For instance, the Cramer-Shoup cryptosystem and signature scheme relies on the "Strong RSA Assumption" [10,1], which allows the adversary to himself choose an exponent $e > 1$. A natural question would be whether factoring is equivalent to solving strong RSA using a GRA. It is not clear whether this statement is true. The proof of Lemma 6, however, does not work for this case because here e will depend on the input a. As a result, in the proof of Lemma 6, $f(a)$ is not a polynomial in a (because the exponent is not independent of a).

References

1. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
2. Boneh, D., Lipton, R.: Black box fields and their application to cryptography. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 283–297. Springer, Heidelberg (1996)
3. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (1998)
4. Brown, D.R.L.: Breaking RSA may be as difficult as factoring. In: Cryptology ePrint Archive, Report 205/380 (2006)
5. Childs, L.: A concrete introduction to higher algebra. Springer, New York (1992)
6. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
7. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. In: 6th ACM Conference on Computer and Communications Security, pp. 46–52 (1999)
8. Damgård, I.B., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 256–271. Springer, Heidelberg (2002)
9. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
10. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
11. Jager, T.: Generic group algorithms. Master's thesis, Ruhr Universität Bochum (2007)
12. Joux, A., Naccache, D., Thomé, E.: When e-th roots become easier than factoring. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 13–28. Springer, Heidelberg (2007)
13. Leander, G., Rupp, A.: On the equivalence of RSA and factoring regarding generic ring algorithms. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 241–251. Springer, Heidelberg (2006)
14. Lidl, R., Niederreiter, H.: Introduction to finite fields and their applications. Cambridge University Press, Cambridge (1994)
15. Maurer, U.: Towards the equivalence of breaking the diffie-hellman protocol and computing discrete logarithms. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 271–281. Springer, Heidelberg (1994)
16. Maurer, U.: Fast generation of prime numbers and secure public-key cryptographic parameters. Journal of Cryptology 8(3), 123–155 (1995)
17. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)
18. Maurer, U., Wolf, S.: The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. SIAM Journal of Computing 28(5), 1689–1721 (1999)

19. Micciancio, D.: The RSA group is pseudo-free. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 387–403. Springer, Heidelberg (2005)
20. Nechaev, V.I.: Complexity of a deterministic algorithm for the discrete logarithm. Mathematical Notes 55(2), 91–101 (1994)
21. Rivest, R.L.: On the notion of pseudo-free groups. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 505–521. Springer, Heidelberg (2004)
22. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM 21, 120–126 (1978)
23. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)

Resettably Secure Computation

Vipul Goyal* and Amit Sahai**

Department of Computer Science, UCLA

Abstract. The notion of resettable zero-knowledge (rZK) was introduced by Canetti, Goldreich, Goldwasser and Micali (FOCS'01) as a strengthening of the classical notion of zero-knowledge. A rZK protocol remains zero-knowledge even if the verifier can reset the prover back to its initial state anytime during the protocol execution and force it to use the same random tape again and again. Following this work, various extensions of this notion were considered for the zero-knowledge and witness indistinguishability functionalities.

In this paper, we initiate the study of resettability for more general functionalities. We first consider the setting of resettable two-party computation where a party (called the user) can reset the other party (called the smartcard) anytime during the protocol execution. After being reset, the smartcard comes back to its original state and thus the user has the opportunity to start interacting with it again (knowing that the smartcard will use the same set of random coins). In this setting, we show that it is possible to secure realize *all PPT computable functionalities* under the most natural (simulation based) definition. Thus our results show that in cryptographic protocols, the reliance on randomness and the ability to keep state can be made significantly weaker. Our simulator for the aforementioned resettable two-party computation protocol (inherently) makes use of non-black box techniques. Second, we provide a construction of *simultaneous* resettable multi-party computation with an honest majority (where the adversary not only controls a minority of parties but is also allowed to reset any number of parties at any point). Interestingly, all our results are in the plain model.

1 Introduction

The notion of resettable zero-knowledge (rZK) was introduced by Canetti et al [CGGM00] with a motivation towards obtaining zero-knowledge protocols for highly adversarial environments. In rZK, the verifier is given the additional power that anytime during the protocol execution, it can "reset" the prover back to its initial state thus restarting the prover with the same configuration and coin

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* Research supported in part from Amit Sahai's grants and a Microsoft Graduate Research Fellowship.

** Research supported in part from NSF grants 0627781, 0716389, 0456717, and 0205594, a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Alfred P. Sloan Foundation Fellowship, and an Okawa Foundation Research Grant.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 54–71, 2009.

tosses. This notion is motivated by several natural questions. Firstly, it address the question: "is zero-knowledge possible when the prover uses the *same random coins* in more than one execution?" and (surprisingly) gives a positive answer to it. Secondly, it shows that zero-knowledge protocols can be securely implemented by devices which can neither reliably keep state nor toss coins online. An example of such a device might be a resettable stateless "smartcard" with secure hardware (which can be reset, e.g., by switching off its power) [CGGM00]. Thus, rZK can be viewed as "zero-knowledge protocols for stateless devices". Canetti et al [CGGM00] provide rZK proof system (for NP) with non-constant number of rounds and resettable witness indistinguishable (rWI) proof system with a constant number of rounds.

Canetti et al [CGGM00] observed that since the prover can be reset, an adversarial verifier can actually achieve the effect of interacting with the prover *concurrently* in several sessions, thus in particular implying concurrent zero knowledge [DNS98]. In more detail, the verifier can start a session with the prover and while the interaction is still in progress, can reset the prover to start another interaction. It could reset the prover again in the middle to that interaction and *come back to the previous interaction* by just running the protocol identically up to the point where it reset the prover before. Thus, the verifier gets the flexibility of choosing its messages in one interaction based on the messages of the prover in some other interaction. In other words, the verifier can essentially interact with the prover in various sessions and can interleave these sessions as it pleases. Thus, a rZK protocol is a concurrent ZK protocol as well [CGGM00]. Following this work, research on improving the round complexity of concurrent ZK also led to rZK with improved round complexity [KP01, PRS02].

Subsequent to the introduction of this notion, various extensions were considered. Barak, Goldreich, Goldwasser and Lindell [BGGL01] studied *resettably-sound* ZK (rsZK) where it is the verifier that can be reset by the prover. The main challenge is to design a ZK protocol which retains its soundness even when verifier uses the same random coins in multiple executions. Relying the non black-box techniques of Barak [Bar01], Barak et al were able to construct constant round rsZK arguments. These rsZK arguments were also used to construct resettable zero-knowledge arguments of knowledge [BGGL01]. An open question raised by [BGGL01] is: do there exist ZK protocols where both the prover and the verifier are resettable by each other? While this still remains an open problem, partial progress was made in [DL07]. The notion of resettability was also studied extensively in the bare public key model introduced by [CGGM00] with a goal of obtaining more efficient protocols (see [YZ07] and the references therein).

Going Beyond ZK – Our Contributions. Common to all the prior work on the notion of resettability is that they consider only the zero-knowledge (or closely related) functionality. This raises the following natural question:

"Do there exist other classes of functionalities for which resettably secure protocols can be obtained?"

In other words, do there exist protocols for other tasks such that the "security" is retained even if one of parties participating in the protocol can be reset by the other one? We initiate the study of general resettable computation and answer the above question in the affirmative.

–**Resettable Two-Party Computation.** We prove a general completeness theorem for resettable computation showing that it is possible to construct a protocol to securely realize *any PPT computable functionality*. Our results are for the setting where one party (called the user) can reset the other party (called the smartcard) during the protocol execution. We first formalize this notion of resettable two-party computation and give a natural simulation based security definition for it. We then construct a general two-party computation protocol under standard (polynomial time) cryptographic assumption. We in fact give a "resettability compiler" which can compile any semi-honest secure (in the standalone setting) protocol into one that is resettably secure. The simulator for our resettable two-party computation protocol makes use of non-black box techniques. We note that non-black box simulation is inherent since Barak et al [BGGL01] show that it is impossible to obtain rsZK arguments (for languages outside $\mathcal{BPP}$) using black-box simulation and rsZK arguments for **NP** is only a special case of a two-party functionality.

–**Resettable Multi-Party Computation.** Given the above results, two natural questions that arise then are: (a) Do there exists general secure resettable *multi-party* computation protocols (where one or more of the parties are resettable)?, and, (b) Can the construction be extended to handle cases where both parties can potentially reset each other? Towards that end, we first observe that the our construction for resettable two-party computation can be extended using standard techniques to show the existence of resettable multi-party computation (where only one of the parties can be reset) with dishonest majority. Next, we offer a construction of *simultaneous* resettable multi-party computation (where all the parties are resettable) assuming a majority of the parties behave honestly. That is, the adversary not only controls a minority of parties but is also allowed to reset any number of honest parties at any point in the protocol execution. At the core of our construction is a new construction of families of multi-party 1-round (and thus automatically resettable) zero-knowledge arguments of knowledge which are simulation sound [Sah99].

Our results show that in cryptographic protocols, the reliance on randomness and the ability to keep state can be made significantly weaker. We in fact show a simple transformation from any resettably secure protocol (as per our definitions) to a fully stateless one. By this we mean that the party which was resettable in the original protocol need not maintain any state at all in the transformed protocol.

Concurrency vs Resettability. As discussed earlier, if a party can be reset, the other party can actually achieve the effect of interacting with it *concurrently* in several sessions. This fact is the reason for the folklore that a resettably secure protocol should also be concurrently secure (i.e., concurrently self-composable).

However far reaching impossibility results have been proven showing that a large class of functionalities cannot be securely realized [Lin03, Lin04] (even in the fixed roles setting) in the plain model. This stands in sharp contrast to our general positive results for resettable two-party computation.

In resettable two-party computation, an adversarial user already has the power to reset the smartcard and interact with it as many times as it likes. This fact that the number of interactions cannot be controlled is precisely what makes resettable two-party computation possible. In the formulation of our ideal model for defining security, we give the adversarial user the power to interact with the smartcard as many times it likes. Given that an adversarial user is allowed to reset the smartcard in the real model (thus creating new problems), emulating the view of such an adversary in the ideal world is only possible if several interactions with the smartcard are allowed. In other words, we are only allowing the user to do in the ideal world what he is already allowed to do in reality. By giving our ideal adversary such extra power, we are able to construct protocols satisfying our definition in the plain model. In our construction, the number of times the simulator sends the reset request in the ideal world is polynomially related to the number of times the real world adversary sends the reset request. An open problem raised by the current work is to design a construction having a *precise simulation* [MP06] with respect to the number of outputs.

Combining our results with the impossibility results of Lindell [Lin03, Lin04], we get a first separation between the notions of resetability and concurrency. That is, a resettably secure protocol (as per our definitions) is not always a concurrently secure protocol (even for fixed roles) and vice versa (this direction is implicit in [CGGM00]). In fact there exists a large class of functionalities for which concurrently self-composable protocols do not exist but resettably secure protocols do (as we show that they exist for every two-party functionality).

Meaningful Resettable Functionalities. We stress that while the resettable setting is unsuitable for several traditional functionalities, such as Yao's Millionaire function, it remains meaningful and highly non-trivial for a large class of functions. For instance, consider the problem of "conditional blind signatures", where one is willing to sign unknown messages as long as they satisfy some property P. If a resettable device were to use a traditional two-party computation protocol to do this, it might leak full knowledge of the secret signing key; our protocols would ensure that only the power to sign messages satisfying P is given to the holder of the device. In general, functionalities where the output is "cryptographic" in nature, and where the input of the device is a cryptographic key, will be meaningful for resettable devices in a variety of settings. Techniques from the area of privacy-preserving data analysis (see [Dwo08] and the references therein) may also be useful in designing other classes of functions suitable for computation by resettable devices.

Stateless vs Stateful Devices. Our results have interesting implications about the power of stateless devices. Consider a stateless device (holding an input, possibly unable to toss coins online) trying to run a protocol for secure computation of some functionality with a user. Our results show that *stateless devices can run*

secure computation protocols for every task. Of course, stateless devices can only be used when one does not want to limit the number of protocol executions that the device carries out (with a particular input).

What if one does want to limit the number of interactions that the device carries out? We remark that it is also possible to obtain the following "best of both worlds" protocol in such a case. *In case* the adversary is not able to reset the device during the protocol interaction, the protocol provides a traditional security guarantee (i.e., security as per the ideal/real world simulation paradigm, see Canetti [Can00]). However if it turns out that the adversary *was successfully* able to reset the device, the *maximum* the adversary can do is to achieve the effect of running the protocol several times with the device (possibly choosing different input each time).

While "protocols for stateless devices" are naturally useful for weak and inexpensive devices (e.g., smartcard), other applications of such protocols could include making a powerful server (providing services to many clients) stateless to prevent denial of service attacks.

Universally Composable Multi-Party Computation using Tamper Proof Hardware. To show one example of the power of our results, we consider the recent work on obtaining universally composable multi-party computation using tamper proof hardware tokens [Kat07]. As noted before, broad impossibility results have been proven showing that a large class of functionalities cannot be UC securely realized in the plain model [CF01, CKL06]. These severe impossibility results motivated the study of other models involving some sort of *trusted* setup assumptions (assuming a trusted third party), where general positive results can be obtained. To avoid these trust assumptions (while still maintaining feasibility of protocols), Katz recently proposed using a *physical setup*. In his model, the physical setup phase includes the parties exchanging tamper proof hardware tokens implementing some functionality.

The security of the construction in [Kat07] relies on the ability of the tamper-resistant hardware to maintain state (even when, for example, the power supply is cut off). In particular, the parties need to execute a four-round coin flipping protocol with the tamper-resistant hardware. Using our techniques, one can immediately relax this requirement and make the token completely stateless. In particular, we can apply our compiler to the coin flipping protocol in [Kat07] and obtain a new construction where the token, when fed with an input x, only outputs $f(x)$ for a fixed f and halts. A construction having such a property was first obtained recently by Chandran et al [CGS08] by relying on techniques that are very specific to the setting of UC secure computation with tamper proof hardware. However our construction has an added advantage that a possibly adversarial token does not learn any information about the input of the honest party using it (and hence the security is retained even when the adversarial creator of the token "recaptures" it at a later point of time). This is leads to the first construction of UC secure computation with stateless and recapturable tokens. On the downside, as opposed to [CGS08], the security of this construction would

rely on the adversary "knowing" the code of the tokens which it distributes to the honest parties (see [CGS08] for more details).

Open Problems. The main question left open by the current work is: "Do there exist two-party and multi-party computation protocols in the *dishonest majority* setting where more than one party is resettable?". Eventually, one would like to construct simultaneous resettable multi-party computation where the adversary can control any number of parties and can reset any number of honest parties at any point (or show that such protocols cannot exist). The apparent obstacle to making any progress towards answering the above questions is the long-standing problem of constructing a simultaneous resettable zero-knowledge argument (first mentioned as an open problem in the work of Barak et al [BGGL01]). We recently settled this conjecture affirmatively in [GS08].

2 The Resettable Ideal Model

2.1 Resettable Two-Party Computation

Informally speaking, our model for resettable two-party computation is as follows. We consider a smartcard $\mathbb{S}$ holding several inputs $x_1^1, \ldots, x_1^{num}$ and random tapes $\omega_1^1, \ldots, \omega_1^{num}$. We denote by X_1 the full input and randomness vector (i.e., the concatenation of all the inputs and random tapes) held by $\mathbb{S}$. The ith *incarnation* of the smartcard $\mathbb{S}$ is a deterministic strategy defined by the pair (x_1^i, ω_1^i) as in [CGGM00]. We stress that while each incarnation has its own random tape, as in [CGGM00], when a particular incarnation is reset, it starts over with the same random tape. Thus, we model different smartcards as the different incarnations. We consider a user $\mathbb{U}$ holding an input x_2 interested in interacting with the ith incarnation of the smartcard $\mathbb{S}$. The user activates the ith incarnation of the smartcard and runs a protocol with it to securely compute a function $f(.,.)$. We do not explicitly define a way of activating the ith incarnation; it could either be through physical means or by sending an initial message to $\mathbb{S}$ specifying the incarnation $\mathbb{U}$ would like to interact with. At any point during the protocol execution, the user $\mathbb{U}$ can *reset* the smartcard $\mathbb{S}$ to its initial state, thus, having a chance to start interaction again with any incarnation with a fresh input. At the end of the protocol, both the parties get the output $f(x_1, x_2)$, $x_1 = x_1^i$ where i and x_2 are the incarnation and the inputs which were most recently selected by the user $\mathbb{U}$. We remark that we only consider *one side resettability*, that is, the smartcard $\mathbb{S}$ is not allowed to reset the user $\mathbb{U}$.

To formalize the above requirement and define security, we extend the standard paradigm for defining secure computation. We define an ideal model of computation and a real model of computation, and require that any adversary in the real model can be *emulated* (in the specific sense described below) by an adversary in the ideal model. In a given execution of the protocol we assume that all inputs have length κ, the security parameter. We consider a static adversary which chooses whom to corrupt before execution of the protocol. In our model, both the parties get the same output (the case where parties should get different

outputs can be easily handled using standard techniques). Finally, we consider *computational* security only and therefore restrict our attention to adversaries running in probabilistic polynomial time.

IDEAL MODEL. In the ideal model there is a trusted party which computes the desired functionality based on the inputs handed to it by the players. Then an execution in the ideal model proceeds as follows:

Select incarnation. The user $\mathbb{U}$ sends an incarnation index i to the trusted party which then passes it on to the smartcard $\mathbb{S}$.

Inputs. The smartcard $\mathbb{S}$ has input x_1 while the user $\mathbb{U}$ has input x_2.

Send inputs to trusted party. Both $\mathbb{S}$ and $\mathbb{U}$ send their inputs to the trusted party. An honest party always sends its real inputs to the trusted party. A corrupted party, on the other hand, may decide to send modified value to the trusted party.

Trusted party computes the result. The trusted party sets the result to be $f(x_1, x_2)$. It generates and uses uniform random coin if required for the computation of f.

Trusted party sends results to adversary. The trusted party sends the result $f(x_1, x_2)$ to either $\mathbb{S}$ or $\mathbb{U}$ depending upon whoever is the adversary.

Trusted party sends results to honest players. The adversary, depending on its view up to this point, does the following. It either sends the *abort* signal in which case the trusted party sends $\perp$ to the honest party. Or it could signal the trusted party to *continue* in which case the trusted party sends the result $f(x_1, x_2)$ to the honest party.

Reset ideal world at any point. In case the user $\mathbb{U}$ is the adversary, during the execution of any of the above steps, it can send the signal *reset* to the trusted party. In that case, the trusted party sends *reset* to the smartcard $\mathbb{S}$ and the ideal world comes back to the *select incarnation* stage.

Outputs. An honest party always outputs the response it received from the trusted party. The adversary outputs an arbitrary function of its entire view throughout the execution of the protocol.

For a given adversary $\mathcal{A}$, the *execution of f in the ideal model* on X_1, x_2 is defined as the output of the honest parties along with the output of the adversary resulting from the process above. It is denoted by $\text{IDEAL}_{f,\mathcal{A}}(X_1, x_2)$.

REAL MODEL. An honest party follows all instructions of the prescribed protocol, while a adversarial party may behave arbitrarily. If the user $\mathbb{U}$ is the adversarial party, it can *reset* the smartcard $\mathbb{S}$ at any point during the protocol execution. After getting reset, $\mathbb{S}$ comes back to its original state which it was in when starting the protocol execution thus allowing $\mathbb{U}$ to choose a fresh input and start interaction again with any incarnation of the smartcard $\mathbb{S}$. At the conclusion of the protocol, an honest party computes its output as prescribed by the protocol. Without loss of generality, we assume the adversary outputs exactly its entire view of the execution of the protocol.

For a given adversary $\mathcal{B}$ and protocol Σ for resettably computing f, the *execution of Σ in the real model* on X_1, x_2 (denoted $\text{REAL}_{\Sigma,\mathcal{B}}(X_1, x_2)$) is defined as

the output of the honest parties along with the output of the adversary resulting from the above process.

Having defined these models, we now define what is meant by a resettable two-party computation protocol. By *probabilistic polynomial time* (PPT), we mean a probabilistic Turing machine with non-uniform advice whose running time is bounded by a polynomial in the security parameter κ. By *expected probabilistic polynomial time* (EPPT), we mean a Turing machine whose *expected* running time is bounded by some polynomial, for *all* inputs.

Definition 1. *Let f and Σ be as above. Protocol Σ is a* secure protocol for computing *f if for every* PPT *adversary $\mathcal{A}$ corrupting either of the two players in the real model, there exists an* EPPT *adversary $\mathcal{S}$ corrupting that player in the ideal model, such that:*

$$\{\text{IDEAL}_{f,\mathcal{S}}(X_1, x_2)\}_{(X_1,x_2)\in(\{0,1\}^*)^2} \stackrel{c}{\equiv} \{\text{REAL}_{\Sigma,\mathcal{A}}(X_1, x_2)\}_{(X_1,x_2)\in(\{0,1\}^*)^2}.$$

Our real model translates to the so called *multiple incarnation non-interleaving* setting in the terminology of [CGGM00]. This setting was shown to be equivalent to the *multiple incarnation interleaving* setting for the case of zero-knowledge and their proof can be extended to the general case as well. In other words, a protocol Σ which is secure when the user $\mathbb{U}$ is allowed only to interact with one incarnation at a time remain secure even if $\mathbb{U}$ is allowed to *concurrently* interact with any number of incarnation simultaneously. For simplicity of exposition, we only consider the setting when the inputs of smartcard $\mathbb{S}$ are all fixed in advance (while $\mathbb{S}$ is acting as honest party in the protocol). However we remark that our protocols also work for the more general case when the inputs of $\mathbb{S}$ are adaptively chosen possibly based on the outputs in the previous protocol executions. More details regarding these issues will be provided in the full version.

2.2 Simultaneous Resettable Multi-party Computation

For lack of space, we defer the model for this case to the full version of this paper. The main changes from the two-party case is that we consider an adversary who controls a minority of the parties and can reset any number of honest parties at any point during the protocol.

2.3 Extensions

In this section, we informally describe two extensions which can be applied to our constructions proven secure as per our definitions. More formal details will be provided in the full version.

Going from Resettable to Stateless. Any protocol which is resettably secure can be transformed to a stateless protocol using relatively standard techniques. In other words, the parties which were allowed to be resettable in the original protocol need not maintain any state at all in the transformed protocol. By a

stateless device we mean that the device only supports a "request-reply" interaction (i.e., the device just outputs $f(x)$ when fed with x for some fixed f). We describe the case of two party first assuming both the parties are resettable (the case where one party is resettable is only simpler). Let we have parties P_1 and P_2 participating in the original resettably secure protocol Σ. Now we define a protocol Σ' having parties P_1' and P_2'. Each of these parties will have a secret key of a CCA-2 secure encryption scheme and a secret MAC key. The party P_1' computes the first message to be sent in the protocol Σ' by running P_1 internally. However it then sends to P_2' not only the computed message but also an encryption of the *current state* of P_1 and a MAC on it. Party P_2' similarly computes the reply by feeding the received message to P_2 and sends to P_1' not only the computed reply but also (a) the received encrypted state of P_1 and the MAC, and, (b) an encryption of the current state of P_2 and a MAC on it using its own keys. Thus for the next round, P_1' can decrypt, verify and *load* the received state into P_1, feed it the incoming reply and then compute the next outgoing message. P_2 can similarly continue with the protocol. The case of multi-party protocols can also be handled by first transforming the given protocol into one where only *one party* sends a message in any given round and then applying ideas similar to the one for the two party case to this resulting protocol.

Getting the Best of Both Worlds. One might ask the question: is it possible to have a single protocol such that *in case* the adversary is not able to reset the device, the protocol provides a traditional security guarantee (i.e., security as per the ideal/real world simulation paradigm, see Canetti [Can00]). However if it turns out that the adversary *is successfully* able to reset the device, the protocol still provides security as per the resettable ideal model definition presented above. We remark that it is easy to transform both our constructions into ones which provide such a best of both worlds guarantee (however we do not know if our transformation works for all constructions). We build a counter into the device which gets incremented with every protocol execution (whether successfully completed or not). The randomness used by the device for a protocol execution comes from the application of a PRF to the current counter value. This guarantees that in case the device *is* able to keep such a counter successfully, the randomness used in each execution is fresh and independent of others. Thus, it is easy to show that one can use a significantly simpler simulator (which can only handle standalone executions) to prove security of our constructions in such a setting.

3 Building Blocks

Our protocols make use of the following building blocks: a commitment scheme COM based on one way permutations, computational zero-knowledge proofs and proofs of knowledge, zaps [DN00], resettable sound zero-knowledge arguments [BGGL01] and the PRS concurrent zero-knowledge preamble [PRS02].

4 Resettable Two-Party Computation

4.1 The Construction

We now describe how to transform any given two party protocol Π (which is only semi-honest secure) into a resettably secure protocol Σ. Prior to the beginning of the protocol, we assume that the smartcard $\mathbb{S}$ and the user $\mathbb{U}$ have agreed upon the incarnation of $\mathbb{S}$ for the protocol. Each incarnation of the smartcard $\mathbb{S}$ has its own independent random tape. We assume that the private inputs to $\mathbb{S}$ and $\mathbb{U}$ in the protocol Π are x_1 and x_2 respectively. The smartcard $\mathbb{S}$ denotes the party which can be reset by the other party $\mathbb{U}$ in the protocol Σ. We view the random tape of the smartcard as a tuple (G, R_{rs}). Here G denotes the description of a function $G : \{0,1\}^{\leq \text{poly}(\kappa)} \rightarrow \{0,1\}^{\text{poly}(\kappa)}$ taken from an ensemble of pseudorandom functions and R_{rs} denotes a random string which $\mathbb{S}$ will use while acting as a verifier of a *resettable sound zero-knowledge* argument. Let R denote the uniform distribution. The protocol proceeds as follows.

PRS Preamble Phase

1. $\mathbb{U} \rightarrow \mathbb{S}$: Generate $r_2 \stackrel{\$}{\leftarrow} R$ and let $\beta = (x_2, r_2)$. Here r_2 is the randomness to be used (after coin flipping with $\mathbb{S}$) by the user $\mathbb{U}$ at various stages of the protocol Σ (including to carry out the protocol Π) as explained later on. We assume that r_2 is of sufficient size to allow $\mathbb{U}$ to execute all such stages. Generate random shares $\{\beta^0_{i,\ell}\}^k_{i,\ell=1}, \{\beta^1_{i,\ell}\}^k_{i,\ell=1}$ such that $\beta^0_{i,\ell} \oplus \beta^1_{i,\ell} = \beta$ for every i, ℓ. Using the commitment scheme COM, commit to all these shares. Denote these commitments by $\{B^0_{i,\ell}\}^k_{i,\ell=1}, \{B^1_{i,\ell}\}^k_{i,\ell=1}$.
   Let msg be the concatenation of all these commitment strings, i.e., $msg = B^0_{1,1}||\ldots||B^0_{k,k}\,||B^1_{1,1}||\ldots||B^1_{k,k}$. We call msg to be the *determining message* of this session (since it commits the user $\mathbb{U}$ to its input and randomness). The random tape used by the smartcard $\mathbb{S}$ to carry out rest of the protocol Σ (except when acting as a verifier of a resettable sound ZK argument) will be determined by the application of the pseudorandom function G to the determining message msg. Again, we assume that $G(msg)$ is of sufficient size to allow the execution of all the steps.
2. $\mathbb{U} \leftrightarrow \mathbb{S}$: The user $\mathbb{U}$ and the smartcard $\mathbb{S}$ will now use a resettable-sound zero-knowledge argument system (rsP, rsV) (relying a non-black box simulator [BGGL01]). $\mathbb{U}$ emulates the prover rsP and proves the following statement to the resettable verifier rsV (emulated by $\mathbb{S}$): the above PRS commit phase is a *valid commit phase* In other words, there exist values $\hat{\beta}$, $\{\hat{\beta}^0_{i,\ell}\}^k_{i,\ell=1}$, $\{\hat{\beta}^1_{i,\ell}\}^k_{i,\ell=1}$ such that (a) $\hat{\beta}^0_{i,\ell} \oplus \hat{\beta}^1_{i,\ell} = \hat{\beta}$ for every i, ℓ, and, (b) Commitments $\{B^0_{i,\ell}\}^k_{i,\ell=1}, \{B^1_{i,\ell}\}^k_{i,\ell=1}$ can be decommitted to $\{\hat{\beta}^0_{i,\ell}\}^k_{i,\ell=1}, \{\hat{\beta}^1_{i,\ell}\}^k_{i,\ell=1}$.
   The user $\mathbb{U}$ uses a fresh (uncommitted) random tape for emulation of the prover rsP. The random tape used by $\mathbb{S}$ to emulate the resettable verifier rsV comes from R_{rs}. Going forward, this will be the case with all the resettable sound zero-knowledge arguments in our protocol Σ.

3. For $\ell = 1, \ldots k$:
   (a) $\mathbb{S} \rightarrow \mathbb{U}$: Send challenge bits $b_{1,\ell}, \ldots, b_{k,\ell} \stackrel{\$}{\leftarrow} \{0,1\}^k$.
   (b) $\mathbb{U} \rightarrow \mathbb{S}$: Decommit to $B_{1,\ell}^{b_{1,\ell}}, \ldots, B_{k,\ell}^{b_{k,\ell}}$.
   The random tape required by $\mathbb{S}$ to generate the above challenge bits comes from $G(msg)$.
4. $\mathbb{S} \rightarrow \mathbb{U}$: Since the underlying protocol Π is secure only against semi-honest adversaries, the random coins used by each party are required to be unbiased. Hence $\mathbb{S}$ generates $r_2' \stackrel{\$}{\leftarrow} R$ (using the random tape from $G(msg)$) and sends it to $\mathbb{U}$. Define $r_2'' = r_2 \oplus r_2'$. Now r_2'' is the randomness which will be used by $\mathbb{U}$ for carrying out the protocol Π (among other things).

Smartcard Input Commitment Phase

1. $\mathbb{S} \rightarrow \mathbb{U}$: Generate a string $r_1 \stackrel{\$}{\leftarrow} R$ and let $\alpha = (x_1, r_1)$. Commit to α using the commitment scheme COM and denote the commitment string by A. The random tape required to generate the string r_1 and to compute the commitment A comes from $G(msg)$.
2. $\mathbb{S} \leftrightarrow \mathbb{U}$: Now $\mathbb{S}$ has to give a proof of knowledge of the opening of the commitment A to $\mathbb{U}$. In other words, $\mathbb{S}$ has to prove that it knows a value $\hat{\alpha} = (\hat{x_1}, \hat{r_1})$ such that the commitment A can be decommitted to $\hat{\alpha}$. This proof is given as follows. $\mathbb{S}$ and $\mathbb{U}$ use an ordinary computational zero-knowledge proof of knowledge system (P_{pok}, V_{pok}) where $\mathbb{S}$ and $\mathbb{U}$ emulates the prover P_{pok} (proving the above statement) and the verifier V_{pok} respectively. However the random tape used by $\mathbb{U}$ to emulate the verifier V_{pok} is required to come from r_2''. To achieve this, the interaction proceeds as follows. Let t'_{pok} be the number of rounds in (P_{pok}, V_{pok}) where a round is defined to have a message from P_{pok} to V_{pok} followed by a reply from V_{pok} to P_{pok}. For $j = 1, \ldots t'_{pok}$:
   - $\mathbb{S} \rightarrow \mathbb{U}$: $\mathbb{S}$ sends the next prover message computed as per the system (P_{pok}, V_{pok}). The random tape used by $\mathbb{S}$ to emulate P_{pok} comes from $G(msg)$.
   - $\mathbb{U} \rightarrow \mathbb{S}$: $\mathbb{U}$ sends the next verifier message computed as per the system (P_{pok}, V_{pok}) *using randomness* r_2''.
   - $\mathbb{U} \leftrightarrow \mathbb{S}$: $\mathbb{U}$ and $\mathbb{S}$ now execute a resettable-sound zero-knowledge argument where $\mathbb{U}$ emulates the prover rsP and proves the following statement to rsV emulated by $\mathbb{S}$: the PRS commit phase has a major decommitment $\hat{\beta} = (\hat{x_2}, \hat{r_2})$ such that the sent verifier message is consistent with the randomness $\hat{r_2} \oplus r_2'$ (where r_2' was as sent by $\mathbb{S}$ in the PRS preamble phase).

   The above system can be seen as a *resettable zero-knowledge argument of knowledge* system [BGGL01]. However when used in our context as above, the simulator of this system will be straightline.
3. $\mathbb{U} \rightarrow \mathbb{S}$: $\mathbb{U}$ generates $r_1' \stackrel{\$}{\leftarrow} R$ using the random tape r_2'' and sends it to $\mathbb{S}$. Define $r_1'' = r_1 \oplus r_1'$. Now r_1'' is the randomness which will be used by $\mathbb{S}$ for carrying out the protocol Π.

4. $\mathbb{U} \leftrightarrow \mathbb{S}$: $\mathbb{U}$ and $\mathbb{S}$ now execute a resettable-sound zero-knowledge argument. $\mathbb{U}$ emulates the prover rsP and proves the following statement to rsV emulated by $\mathbb{S}$: the PRS commit phase has a major decommitment $\hat{\beta} = (\hat{x_2}, \hat{r_2})$ such that message r'_1 is consistent with the randomness $\hat{r_2} \oplus r'_2$.

Secure Computation Phase

Let the underlying protocol Π have t rounds[1] where one round is defined to have a message from $\mathbb{S}$ to $\mathbb{U}$ followed by a reply from $\mathbb{U}$ to $\mathbb{S}$. Let transcript T_1^j (resp. T_2^j) be defined to contain all the messages exchanged between $\mathbb{S}$ and $\mathbb{U}$ before the point party $\mathbb{S}$ (resp. $\mathbb{U}$) is supposed to send a message in round j. Now, each message sent by either party in the protocol Π is compiled into a *message block* in Σ. For $j = 1, \ldots t$:

1. $\mathbb{S} \rightarrow \mathbb{U}$: $\mathbb{S}$ sends the next message m_1^j $(= \Pi(T_1^j, x_1, r''_1))$ as per the protocol Π. Now $\mathbb{S}$ has to prove to $\mathbb{U}$ that the sent message m_1^j was honestly generated using input x_1 and randomness r''_1. In other words, $\mathbb{S}$ has to prove the following statement: there exist a value $\hat{\alpha} = (\hat{x_1}, \hat{r_1})$ such that: (a) the message m_1^j is consistent with the input $\hat{x_1}$ and the randomness $\hat{r_1} \oplus r'_1$ (i.e., $m_1^j = \Pi(T_1^j, \hat{x}, \hat{r_1} \oplus r'_1)$), and, (b) commitment A can be decommitted to $\hat{\alpha}$. This proof is given as follows. $\mathbb{S}$ and $\mathbb{U}$ use an ordinary computational zero-knowledge proof system (P, V) where $\mathbb{S}$ and $\mathbb{U}$ emulates the prover P (proving the above statement) and the verifier V respectively. However the random tape used by $\mathbb{U}$ to emulate the verifier V is required to come from r''_2. To achieve this, the interaction proceeds as follows. Let t' be the number of rounds in (P, V) where a round is defined to have a message from P to V followed by a reply from V to P. For $j' = 1, \ldots t'$:
   - $\mathbb{S} \rightarrow \mathbb{U}$: $\mathbb{S}$ sends the next prover message computed as per the system (P, V). The random tape used by $\mathbb{S}$ to emulate P comes from $G(msg)$.
   - $\mathbb{U} \rightarrow \mathbb{S}$: $\mathbb{U}$ sends the next verifier message computed as per the system (P, V) *using randomness* r''_2.
   - $\mathbb{U} \leftrightarrow \mathbb{S}$: $\mathbb{U}$ and $\mathbb{S}$ now execute a resettable-sound zero-knowledge argument where $\mathbb{U}$ emulates the prover rsP and proves the following statement to rsV emulated by $\mathbb{S}$: the PRS commit phase has a major decommitment $\hat{\beta} = (\hat{x_2}, \hat{r_2})$ such that the sent verifier message is consistent with the randomness $\hat{r_2} \oplus r'_2$.

   To summarize, the random tape used by $\mathbb{U}$ to emulate the verifier V is required to be committed in advance. However $\mathbb{S}$ is free to use any random tape while emulating the prover P (although in the interest of its own security, $\mathbb{S}$ is instructed to use randomness from $G(msg)$.
2. $\mathbb{U}$ sends the next message m_2^j $(= \Pi(T_2^j, x_2, r''_2))$ as per the protocol Π. $\mathbb{U}$ and $\mathbb{S}$ now execute a resettable-sound zero-knowledge argument where $\mathbb{U}$

[1] This assumption is only made for simplicity of exposition. It is easy to extend our construction to handle protocols whose round complexity is not upper bounded by a fixed polynomial.

emulates the prover rsP and proves to $\mathbb{S}$ that m_2^j was honestly generated using input x_2 and randomness r_2''. More precisely, $\mathbb{U}$ proves the following statement: the PRS commit phase has a major decommitment $\hat{\beta} = (\hat{x_2}, \hat{r_2})$ such that m_2^j is consistent with the input $\hat{x_2}$ and the randomness $\hat{r_2} \oplus r_2'$.

This completes the description of the protocol Σ. The usage of randomness in the above protocol can be summarized as follows. After sending the very first message (i.e., the determining message msg), the only fresh randomness that can be used by the user $\mathbb{U}$ is while emulating the prover rsP of resettable sound zero-knowledge arguments. The smartcard $\mathbb{S}$ is essentially "free" to use any randomness it wants (except while computing messages of the underlying protocol Π). An honest $\mathbb{S}$ always sets its random tape to $G(msg)$ to carry out the protocol Σ .

At the end of above protocol Σ, both the parties will hold the desired output. We stress that we require only standard (standalone) semi-honest security from the underlying protocol Π. Thus, when we set the underlying protocol Π to be the constant round two-party computation protocol of Yao [Yao86], the resulting protocol Σ has k $(= \omega(\log \kappa))$ rounds. To obtain a constant round protocol, the first step would be the construction of a concurrent zero-knowledge argument system in a constant number of rounds. We also remark that the above resettable two-party computation also implies (under standard assumptions) resettable multi-party computation (where only one of the parties can be reset) with dishonest majority. The construction for resettable multi-party computation can be obtained using standard techniques from the two-party one (i.e., the $n-1$ "non-resettable" parties will use a regular multi-party computation protocol [GMW87] among them to emulate a single party holding $n-1$ inputs).

We prove that the protocol Σ is a resettable two-party computation protocol by proving the following two theorems; the proofs are deferred to the full version of this paper.

Theorem 1 (Security Against a Malicious $\mathbb{U}^*$). *The compiled protocol Σ is secure against a malicious $\mathbb{U}^*$.*

Theorem 2 (Security Against a Malicious $\mathbb{S}^*$). *The compiled protocol Σ is secure against a malicious $\mathbb{S}^*$.*

5 Simultaneous Resettable Multi-party Computation with Honest Majority

5.1 The Construction

We now describe how to transform any given protocol Π (which is only semi-honest secure) into a simultaneous resettably secure protocol Σ with honest majority. We assume n parties $P_1, \ldots, P_n$ where a majority of the parties behave honestly. All the parties are "resettable", or in other words, the adversarial parties can reset any number of honest parties at any time during the protocol

execution. We assume that before the protocol starts, the parties have agreed upon which incarnation will be used by which party. The private inputs of parties $P_1, \ldots, P_n$ are denoted by $x_1, \ldots, x_n$ respectively. Let R denote the uniform distribution. The protocol Σ proceeds as follows.

Input Commitment Phase

Each party P_i does the following computations. Any randomness required for these computations comes from the random tape of (appropriate incarnation of) P_i which is potentially reusable in other sessions.

- Generate a function $G_i : \{0,1\}^{\leq \text{poly}(\kappa)} \rightarrow \{0,1\}^{\text{poly}(\kappa)}$ randomly from an ensemble of pseudorandom functions and let $\alpha_i = (x_i, G_i)$. Compute a commitment to α_i using the commitment scheme COM and denote it by A_i.
- Generate the first verifier message $Z_i \leftarrow f_{zaps}(\kappa)$ of a zap system.
- Generate a pair (PK_i, SK_i) of public and secret keys using the key generation algorithm of a semantically secure public key encryption system having perfect completeness.
- Generate n strings $a_i^1, \ldots, a_i^n$ from the domain of a one way function F. For all j, compute $b_i^j = F(a_i^j)$.

Additionally, we assume that a party P_i has a random tape $R_{i,zkv}$ which it uses for the verification of messages of a 1 round ZKAOK system as we explain later on. P_i now broadcasts the values $A_i, Z_i, PK_i, b_i^1, \ldots, b_i^n$. Let the string broadcast (i.e., $A_i||Z_i||PK_i||\ b_i^1||\ldots||b_i^n$) be denoted by msg_i. The string msg_i is called the *determining message* of party P_i for this session. Note that since a party P_i may have to reuse its random tape, the determining message msg_i may be identical across various protocol executions.

The random tape used by P_i to carry out rest of the protocol Σ (except for the verification of the messages of the 1 round ZKAOK system) will be determined by the application of the pseudorandom function G_i to the (concatenation of) determining messages of all other parties. That is, denote $R_i = G_i(msg_1||\ldots||msg_{i-1}||msg_{i+1}||\ldots||msg_n)$. Now R_i serves as the random tape of P_i for the rest of the protocol. We assume that R_i is of sufficient size to allow the execution of all the steps.

Construction of a 1-round zero-knowledge argument of knowledge. We now describe the construction of a family of 1-round zero-knowledge argument of knowledge (ZKAOK) systems. The (i,j)th argument system is used by party P_i to prove statements to party P_j. Additionally, the argument systems (i,j) and (k,ℓ), where $i \neq k$, are simulation sound w.r.t. each other. To prove a statement $x \in L$ to party P_j, a party P_i holding the witness w (for the given witness relation) proceeds as follows:

- P_i breaks the witness w into n shares $w_1, \ldots, w_n$ using the Shamir threshold secret sharing scheme [Sha79] such that a majority of the shares are sufficient to reconstruct the witness w. For all k, P_i encrypts w_k under the public key PK_k. Let the ciphertext be denoted by C_k. P_i now broadcasts all the n ciphertexts so generated.

- P_i finally generates and sends the prover message of the zap system (acting on the verifier message Z_j) proving that one of the following statements is true:
  1. The ciphertexts $C_1, \ldots, C_n$ represent the encryption of shares of a valid witness. More precisely, there exists strings $\hat{w_1}, \ldots, \hat{w_n}$ such that: (a) for all k, C_k is a valid encryption of $\hat{w_k}$, and, (b) $\hat{w_1}, \ldots, \hat{w_n}$ are valid shares of a single string $\hat{w}$ as per the Shamir secret sharing scheme, and, (c) $\hat{w}$ is a valid witness for the witness relation (i.e., $x \in L$).
  2. The ciphertexts $C_1, \ldots, C_n$ represent the encryption of shares of a majority of preimages of the strings $b_1^i, \ldots, b_n^i$ under the one way function F. More precisely, there exists strings $\hat{s_1}, \ldots, \hat{s_n}$ such that: (a) for all k, C_k is a valid encryption of $\hat{s_k}$, and, (b) $\hat{s_1}, \ldots, \hat{s_n}$ are valid shares of a single string $\hat{s}$ as per the Shamir secret sharing scheme, and, (c) $\hat{s} = (\hat{a_1}^i, \ldots, \hat{a_n}^i)$ and there exists a set S_{maj} of indices such that $|S_{maj}| > n/2$ and for all $\ell \in S_{maj}$, $b_\ell^i = F(\hat{a_\ell}^i)$.

Thus, we have a *trapdoor condition* which allows a party P_i to give a simulated argument using the preimages of $b_1^i, \ldots, b_n^i$. Note that the "trapdoor" for each party is "independent". That is, informally speaking, even given the preimages of strings $b_1^i, \ldots, b_n^i$, a party P_j with $j \neq i$ will be unable to give a simulated argument.

Coin Flipping Phase

Since the underlying protocol Π is secure only against semi-honest adversaries, the random coins used by each party in Π are required to be unbiased. Hence the parties run a 2 round coin flipping phase to generate a long unbiased public random string as given below. As noted before, the random tape that a party P_i uses to execute this stage (i.e., generate of random strings, commitment and messages of the 1-round ZKAOK system) comes from R_i. However the random tape (if needed) used by P_i to verify the messages of the ZKAOK system comes from $R_{i,zkv}$.

- In the first round, each party P_i generates $R_i' \stackrel{\$}{\leftarrow} R$ and broadcasts a commitment B_i to R_i' using the commitment scheme COM. For all $j \neq i$, party P_i additionally broadcasts a ZKAOK (using the (i,j)th 1-round ZKAOK system) proving that the commitment B_i was correctly computed using randomness R_i. More precisely, P_i proves that there exists a string $\hat{\alpha_i} = (\hat{x_i}, \hat{G_i})$ such that: (a) the commitment A_i can be decommitted to $\hat{\alpha_i}$, and, (b) the commitment B_i to a random string (and the random string itself) was computed using randomness $\hat{G_i}(msg_1||\ldots||msg_{i-1}||msg_{i+1}||\ldots||msg_n)$. Note that this ZKAOK is given for a specific witness relation such that the witness allows extraction of such a $\hat{\alpha_i}$.
- In the second round, each party P_i broadcasts the committed string R_i' (without providing any decommitment information). For all $j \neq i$, party P_i additionally broadcasts a ZKAOK (using the (i,j)th 1-round ZKAOK system) proving that the commitment B_i can be decommitted to the string

R'_i. Denote $r'_1||\ldots||r'_n = R'_1 \oplus \cdots \oplus R'_n$. At this point, the strings $r'_1, \ldots, r'_n$ are all guaranteed to be random.

Each P_i further privately generates a random r_i. Define $r''_i = r_i \oplus r'_i$. Now r''_i is the randomness that will be used by party P_i to carry out the underlying protocol Π in the next stage.

Secure Computation Phase

Let the underlying protocol Π have t rounds[2] where any number of parties can send a message in any given round. Let transcript T^j be defined to contain all the messages broadcast *before* the round j. Now, for $j = 1, \ldots t$:

1. P_i sends the next message m_i^j $(= \Pi(T^j, x_i, r''_i))$ as per the protocol Π. Note that m_i^j could potentially be $\perp$.
2. For all $k \neq i$, party P_i additionally broadcasts a ZKAOK (using the (i,k)th 1-round ZKAOK system) proving that the sent message m_i^j was honestly generated using input x_i and randomness r''_i. In other words, P_i proves the following statement: there exist a value $\hat{\alpha_i} = (\hat{x_i}, \hat{G_i})$ such that: (a) the message m_i^j is consistent with the input $\hat{x_i}$ and the randomness $\hat{r_i} \oplus r'_i$ (i.e., $m_i^j = \Pi(T^j, \hat{x_i}, \hat{r_i} \oplus r'_i)$) where $\hat{r_i}$ is generated from $\hat{G_i}$, and, (b) commitment A_i can be decommitted to $\hat{\alpha_i}$. As before, the random tape used by P_i for generation and verification of the messages of ZKAOK system comes from R_i and $R_{i,zkv}$ respectively.

This completes the description of the protocol Σ. The usage of randomness in the above protocol can be summarized as follows. After sending the very first message (i.e., the determining message msg_i), the only fresh and uncommitted random tape that *can potentially* be used by a malicious party P_i is for the generation and verification of the 1-round ZKAOK messages (although the honest parties are instructed to use the committed random tape for the *generation* of 1-round ZKAOK messages).

Applying the above transformation to the constant round protocol of Beaver et al [BMR90], we obtain a constant round protocol Σ (secure against a minority of malicious parties). Our protocol is based on computational assumptions; the existence of NIZK (or equivalently, two round zaps [DN00]) to be precise. We note that it is easy to rule out information theoretically secure protocols in our setting very similar to how Barak et al [BGGL01] ruled out resettably-sound zero-knowledge *proofs*. The basic idea is that since an honest party has only a bounded size secret information (i.e., the input and the random tape), an unbounded dishonest party can interact with it several times (by resetting it each time) so as to "almost" learn its input/output behavior (and hence an honest party input "consistent" with that behavior). More details will be provided in the full version.

[2] As before, this assumption is only made for simplicity of exposition.

Let $\mathcal{M}$ be the list of malicious parties. Denote the list of honest parties $\mathcal{H} = \{P_1, \ldots, P_n\} - \mathcal{M}$. We defer the proof the following theorem to the full version of this paper for lack of space.

Theorem 3 (Security Against a Minority of Malicious Parties). *The compiled protocol Σ is secure as per definition 2.2 against the coalition of malicious parties represented by $\mathcal{M}$ as long as $|\mathcal{M}| < n/2$.*

References

[Bar01] Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS, pp. 106–115 (2001)

[BGGL01] Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-sound zero-knowledge and its applications. In: FOCS, pp. 116–125 (2001)

[BMR90] Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: STOC, pp. 503–513. ACM, New York (1990)

[Can00] Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology: the journal of the International Association for Cryptologic Research 13(1), 143–202 (2000)

[CF01] Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

[CGGM00] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC, pp. 235–244 (2000)

[CGS08] Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)

[CKL06] Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. J. Cryptology 19(2), 135–167 (2006)

[DL07] Deng, Y., Lin, D.: Instance-dependent verifiable random functions and their application to simultaneous resettability. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 148–168. Springer, Heidelberg (2007)

[DN00] Dwork, C., Naor, M.: Zaps and their applications. In: FOCS, pp. 283–293 (2000)

[DNS98] Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: STOC, pp. 409–418 (1998)

[Dwo08] Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)

[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC 1987: Proceedings of the 19th annual ACM conference on Theory of computing, pp. 218–229. ACM Press, New York (1987)

[GS08] Goyal, V., Sahai, A.: Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. Cryptology ePrint Archive, Report 2008/545 (2008), `http://eprint.iacr.org/`

[Kat07] Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)

[KP01] Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In: STOC, pp. 560–569 (2001)

[Lin03] Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC, pp. 683–692. ACM Press, New York (2003)

[Lin04] Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)

[MP06] Micali, S., Pass, R.: Local zero knowledge. In: Kleinberg, J.M. (ed.) STOC, pp. 306–315. ACM, New York (2006)

[PRS02] Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS, pp. 366–375 (2002)

[Sah99] Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553 (1999)

[Sha79] Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)

[Yao86] Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167. IEEE, Los Alamitos (1986)

[YZ07] Yung, M., Zhao, Y.: Generic and practical resettable zero-knowledge in the bare public-key model. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 129–147. Springer, Heidelberg (2007)

On the Security Loss in Cryptographic Reductions*

Chi-Jen Lu

Institute of Information Science, Academia Sinica, Taipei, Taiwan
cjlu@iis.sinica.edu.tw

Abstract. Almost all the important cryptographic protocols we have today base their security on unproven assumptions, which all imply $\mathsf{NP} \neq \mathsf{P}$, and thus having unconditional proofs of their security seems far beyond our reach. One research effort then is to identify more basic primitives and prove the security of these protocols by reductions to the security of these primitives. However, in doing so, one often observes some security loss in the form that the security of the protocols is measured against weaker adversaries, e.g., adversaries with a smaller running time. Is such a security loss avoidable? We study two of the most basic cryptographic reductions: hardness amplification of one-way functions and constructing pseudorandom generators from one-way functions. We show that when they are done in a certain black-box way, such a security loss is in fact unavoidable.

1 Introduction

Although we have many protocols today for all kinds of interesting and important cryptographic tasks, almost all of these protocols have their security based on some assumptions. These assumptions all imply $\mathsf{P} \neq \mathsf{NP}$, so having unconditional proofs of their security seems far beyond our reach. One line of research then is to identify the weakest possible assumptions or primitives from which one can build more advanced cryptographic protocols. One such primitive is one-way function (OWF), a function which is easy to compute but hard to invert, with respect to polynomial time computation. It is now known that from a OWF, one can construct other cryptographic primitives such as pseudo-random generator (PRG), pseudo-random function, private-key encryption, bit commitment, zero-knowledge proof, and digital signature. In fact, all these primitives are known to be equivalent in the sense that they can all be built from each other [21,6,9,7,11,18,17,10]. According to [5], these primitives may be categorized as in the world of private cryptography. There are other primitives, including public-key encryption, oblivious transfer, private information retrieval, and key agreement, which may be categorized as in the world of public cryptography. Primitives in the world of public cryptography seem to require a stronger assumption, and it has been shown that trapdoor one-way permutations can be

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* This work supported was in part by the National Science Council under the Grant NSC97-2221-E-001-012-MY3.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 72–87, 2009.

used to build all of them. The relationships among primitives in public cryptography are more complicated, but most of them have been settled [13,12,5,2,14,4,1].

From a theoretical perspective, we seem to have obtained a good understanding of the relationships among these primitives. However, from a practical point of view, there are still issues to be resolved. The first is that even when we can construct one primitive from another, the construction may not be as efficient as we desire. For example, although we can use any one-way function to construct all the primitives in the world of private cryptography, the constructions often do not appear efficient enough to have a practical impact. Can we improve the efficiency of such constructions? Some negatives results have been obtained recently for the tasks of amplifying hardness of OWF [15,16], constructing PRG from OWF [3,20,16], and constructing encryption or signature scheme [3] from (trapdoor) one-way permutation.

The second issue is that when constructing a primitive from another, one often suffers some kind of security loss. For example, although one can construct a PRG from a OWF, the proofs currently available can only guarantee the security of the PRG for weaker adversaries having a smaller running time (or circuit size) than that for OWF. Therefore, if we want to have a PRG with a certain security level, we need to start from a OWF with a much higher security level, which would require a substantial cost to implement and make it less attractive in practice. Similar problems also occur in other constructions, and people have tried to improve these constructions, but with limited success so far. Again, one may wonder whether or not such improvements are indeed impossible. Not much seems to be known, and our goal is to show that such security losses are basically unavoidable. We would like to start from two of the most basic primitives: OWF and PRG, and study the task of hardness amplification for OWF and the task of constructing PRG from OWF.

We say that a function f is ε-hard (to invert) for time t (or size t), if any algorithm running in time t (or circuit of size t) must fail to invert $f(x)$ for at least ε fraction of x. The task of hardness amplification is to transform a function f which is ε-hard for time t into a function $\bar{f}$ which is $(1-\delta)$-hard for time $\bar{t}$, for some small δ and ε. According to [21,8], this is possible with $\bar{t} = t/\gamma_1$, for some $\gamma_1 = ((1/\delta)/\log(1/\varepsilon))^{O(1)}$ (it seems that a more careful analysis can give $\gamma_1 = O((1/\delta)/\log(1/\varepsilon))$). That is, the hardness of the new function $\bar{f}$ is now measured against algorithms with a running time (or circuit size) smaller by a γ_1 factor than that for the initial function f. Therefore, when we want to transform a weakly OWF (with hardness $n^{-O(1)}$) into a strongly OWF (with hardness $1-n^{-\omega(1)}$), we lose a polynomial factor in the running time (or circuit size) of adversaries. We say that a function $g : \{0,1\}^n \to \{0,1\}^m$ is ε-random for time t (or size t) if for any algorithm C running in time t (or circuit of size t), the probabilities of $C(u) = 1$ and $C(g(x)) = 1$, over random u and random x respectively, differ by at most ε. According to [10], one can construct such a function g with $m > n$ (a PRG), which is ε-random for time t/γ_2 (or size t/γ_2), from any function f which is $(1-n^{-\Omega(1)})$-hard for time t (or size t), for some

$\gamma_2 = (n/\varepsilon)^{O(1)}$. From [7], one can have $\gamma_2 = n^{O(1)}/\varepsilon^2$, for the simpler case when $m = n + 1$ and f is a permutation.

We would like to show the impossibility of having a hardness amplification of OWF or a construction of PRG from OWF which can avoid such a loss of security. However, it is not clear how to establish the impossibility of transforming one primitive P to another primitive Q, especially given the possibility that the primitive Q may indeed exist. Therefore, one can only expect to have such impossibility results for a certain restricted types of transformations. Here, we consider transformations which are done in some black-box way.

Black-Box Reductions. The standard notion of black-box transformation from a primitive P to a primitive Q consists of two oracle algorithms $T^{(\cdot)}$ and $R^{(\cdot)}$ satisfying the following two conditions: (1) correctness: for any N that implements P, T^N implements Q, and (2) security: for any N that implements P and for any A that breaks T^N (as an implementation of Q), $R^{A,N}$ breaks N (as an implementation of P).

Although this may look restricted, almost all the known transformations between primitives in cryptography (including those we discussed before) are done in such a black-box way. In this paper, we consider a more general model, in which we drop the first condition and keep only the second one, namely, only the security proof is required to be done in a black-box way. We call this the weakly-black-box model, and note that impossibility results on such a more general model become stronger. We consider two transformations in this model: hardness amplification for OWF and constructing PRG from OWF. In the case of weakly-black-box hardness amplification, there exists an oracle algorithm R (an adversary) such that for any M which breaks the hardness condition of the new function $\bar{f}$, R using M as an oracle can break the hardness condition of the initial function f. In the case of weakly-black-box PRG construction, there exists an oracle algorithm R (an adversary) such that for any D which breaks the randomness condition of the resulting generator g, R using D as an oracle can break the hardness condition of the initial function f. Here we consider the more general case in which R can be non-uniform by allowing it to have an advice string (or seeing R as a collection of circuits with oracle gates), and again this makes our impossibility results stronger.

Our Results. We first consider the task of weakly-black-box hardness amplification for OWF, which transforms ε-hard functions into $(1 - \delta)$-hard functions. Our first two results show that any algorithm R realizing such a hardness amplification must make at least $q_1 = \Omega((1/\delta)/\log(1/\varepsilon))$ queries to the oracle, unless it can use a long advice string. More precisely, our first result shows that for any R which is allowed to make adaptive oracle queries, it must make at least q_1 oracle queries or use some linear-size advice. This implies that when doing hardness amplification in this way and considering adversaries as uniform (or slightly non-uniform) algorithms, one can only guarantee the hardness of the new function $\bar{f}$ against adversaries with a computation time smaller by a q_1 factor, so a security loss of this factor is in fact unavoidable. Our second result shows that

for any R which can only make non-adaptive queries, it must again makes at least q_1 oracle queries or now use an advice of exponential length. This implies that when doing hardness amplification in this way and considering adversaries as non-uniform circuits of some small exponential size, one can only guarantee the hardness of the new function $\bar{f}$ against adversaries with a circuit size smaller by a q_1 factor, so a security loss of this factor is again unavoidable.

We next consider the task of weakly-black-box construction of PRG from OWF, which transforms $(1-\delta)$-hard functions into ε-random functions. Our third and forth results show that any algorithm R realizing such a construction must make at least $q_2 = \Omega(n/\varepsilon^2)$ queries, unless it can use a long advice. More precisely, our third result shows that for any R which is allowed to make adaptive oracle queries, it must make at least q_2 oracle queries or use some linear-size advice. Again, this implies that when constructing PRG in this way and considering adversaries as uniform (or slightly non-uniform) algorithms, a security loss of a q_2 factor is in fact unavoidable. Finally, our forth result shows that for any R which can only make non-adaptive queries, it must again make at least q_2 oracle queries or now use an advice of exponential length. Again, this implies that when constructing PRG in this way and considering adversaries as non-uniform circuits of some small exponential size, a security loss of a q_2 factor is also unavoidable.

We remark that in a different setting, Shaltiel and Viola [19] recently showed that for the task of amplifying the hardness of computing Boolean functions (instead of inverting one-way functions), a security loss in terms of circuit size is also unavoidable when this is done in a black-box way. However, they only considered the case that the oracle algorithm R makes non-adaptive queries, and there seem to be further complications when dealing with inverting one-way functions. On the other hand, our proof (for hardness amplification, with R making non-adaptive queries) is different in spirit from theirs, and our proof can be modified to give an alternative (and perhaps more intuitive) proof to their result.

2 Preliminaries

For $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, 2, \ldots, n\}$ and let $\mathcal{U}_n$ denote the uniform distribution over $\{0,1\}^n$. We will consider the computational model of non-uniform oracle algorithms. For such an algorithm R, let $R^{f;\alpha}$ denote the algorithm R using f as an oracle and α as an advice.

For a many-to-one function $f : \{0,1\}^n \to \{0,1\}^m$ and an algorithm $M : \{0,1\}^m \to \{0,1\}^n$, we say that M inverts $f(x)$, denoted as $M(f(x)) \equiv x$, if $M(f(x)) \in f^{-1}(f(x))$, and we say that M can α-invert f, if

$$\Pr_{x \in \mathcal{U}_n}[M(f(x)) \equiv x] \geq \alpha.$$

For a function $g : \{0,1\}^n \to \{0,1\}^m$ and an algorithm $D : \{0,1\}^m \to \{0,1\}$, we say that D can ε-distinguish g if

$$\left| \Pr_{x \in \mathcal{U}_n}[D(g(x)) = 1] - \Pr_{u \in \mathcal{U}_m}[D(u) = 1] \right| \geq \varepsilon.$$

One can allow M and D to be probabilistic, in which case the probabilities above are taken also over their randomness.

Next, let us introduce two types of black-box transformations which we will study in this paper. Note that in a usual black-box model, both the construction and the security proof are required to be done in a black-box way. Here we consider weaker models which only require the security proof to be done in a black-box way.

Definition 1. *A weakly-black-box hardness amplification from (ε, n, m)-hardness to $(\bar{\varepsilon}, \bar{n}, \bar{m})$-hardness consists of a non-uniform oracle algorithm R satisfying the following condition. For any function $f : \{0,1\}^n \to \{0,1\}^m$, there exists a function $\bar{f} : \{0,1\}^{\bar{n}} \to \{0,1\}^{\bar{m}}$ such that*

- *given any $M : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$ which can $(1-\bar{\varepsilon})$-invert $\bar{f}$, there exists an advice α such that $R^{f,M;\alpha}$ can $(1-\varepsilon)$-invert f.*

Definition 2. *A weakly-black-box transformation from (ε, n, m)-hardness to $(\bar{\varepsilon}, \bar{n}, \bar{m})$-randomness consists of a non-uniform oracle algorithm R satisfying the following condition. For any function $f : \{0,1\}^n \to \{0,1\}^m$, there exists a function $g : \{0,1\}^{\bar{n}} \to \{0,1\}^{\bar{m}}$, with $\bar{m} \geq \bar{n} + 1$, such that*

- *given any $D : \{0,1\}^{\bar{m}} \to \{0,1\}$ which can $\bar{\varepsilon}$-distinguish g, there exists an advice α such that $R^{f,D;\alpha}$ can $(1-\varepsilon)$-invert f.*

In the two definitions above, the oracle algorithm R in general is allowed to make *adaptive* queries, which can depend on the answers from previous queries. We will also consider the case requiring that R only makes *non-adaptive* queries, which do not depend on the answers from previous queries but can depend on the input and the advice.

We will need the following (known) fact that a randomly chosen function is likely to be hard to invert. The proof can be modified from those in, e.g., [3,22], which we omit here.

Lemma 1. *Let c any constant such that $0 < c < 1$, and let C be any non-uniform oracle algorithm which uses an advice of length at most 2^{cn} and makes at most 2^{cn} queries to the oracle. Then there is a constant $c_1 > 0$ such that if we sample a random function $f : \{0,1\}^n \to \{0,1\}^m$,*

$$\Pr_f \left[\exists \alpha : C^{f;\alpha} \text{ can } 2^{-c_1 n}\text{-invert } f\right] \leq 2^{-2^{\Omega(n)}}.$$

Finally, we will rely on the following lemma, which gives a large deviation bound for a sequence of random variables with a sparse dependency relationship. This may have some interest of its own.

Lemma 2. *Suppose $Z_1, \ldots, Z_k$ is a sequence of binary random variables such that for each $i \in [k]$, $\mathrm{E}[Z_i] = \mu_i$ and Z_i is mutually independent of all but $d-1$ other random variables. Then for any even $t \in \mathbb{N}$ and for any $A \in \mathbb{N}$, $\Pr[|\sum_{i\in[k]} Z_i - \sum_{i\in[k]} \mu_i| \geq A] \leq 2(4tdk/A^2)^{t/2}$.*

Due to the space limitation, we omit the proof here. The idea is that $\Pr[|\sum_{i\in[k]} Z_i - \sum_{i\in[k]} \mu_i| \geq A] \leq \mathrm{E}[(\sum_{i\in[k]}(Z_i - \mu_i))^t]/A^t$, and the numerator equals $\sum_{i_1,\ldots,i_t\in[k]} \mathrm{E}[\prod_{i\in\{i_1,\ldots,i_t\}}(Z_i - \mu_i)]$ which have most of the terms equal to zero.

3 Hardness Amplification

In this section, we show that any algorithm R realizing a weakly-black-box hardness amplification must make many queries, unless it can use a long advice. Our first result, Theorem 1 below, shows such a query lower bound for any R which is allowed to use an advice of linear length and to make adaptive queries. We will give the proof in Subsection 3.1.

Theorem 1. *Suppose an algorithm R uses an advice of length $\bar{\ell}$ and realizes a weakly-black-box hardness amplification from (ε, n, m)-hardness to $((1-\delta), \bar{n}, \bar{m})$-hardness, with $2^{-cn} \leq \varepsilon, \delta \leq c$ and $\bar{\ell} \leq cn$ for a small enough constant $c > 0$. Then R must make at least $\Omega((1/\delta)\log(1/\varepsilon))$ oracle queries.*

Our second result, Theorem 2 below, shows a query lower bound for any R which is even allowed an advice of exponential length but can only make non-adaptive queries. We will give the proof in Subsection 3.2.

Theorem 2. *Suppose an algorithm R uses an advice of length ℓ and realizes a weakly-black-box hardness amplification from (ε, n, m)-hardness to $((1-\delta), \bar{n}, \bar{m})$-hardness, with $2^{-cn} \leq \varepsilon, \delta \leq c$ and $\ell \leq 2^{cn}$ for a small enough constant $c > 0$. Then R must make at least $\Omega((1/\delta)\log(1/\varepsilon))$ oracle queries, if it only makes* non-adaptive *queries.*

3.1 Proof of Theorem 1

Consider any non-uniform oracle algorithm R which realizes such a hardness amplification. Assume that R makes at most $q \leq (c_0/\delta)\log(1/\varepsilon)$ oracle queries, for a small enough constant c_0, and we will show that this leads to a contradiction. The basic idea is that when R makes only a small number of queries, it is easy to get confused between some useful oracle $M : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$ (which is correlated with f) and a useless one $\bar{0} : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$ (which is independent of f). Here, we take $\bar{0}$ to be the all-zero function, where $\bar{0}(\bar{y}) = 0^{\bar{n}}$ for any $\bar{y} \in \{0,1\}^{\bar{m}}$. We will first describe a natural approach which will encounter two obstacles, and we will then show how to modify the approach to overcome the obstacles.

First, we would like to pick a function $f : \{0,1\}^n \to \{0,1\}^m$, such that f is hard to invert by $R^{f,\bar{0};\alpha}$ for any advice α, and the corresponding harder function $\bar{f}$ does not map many inputs into a small subset. Its existence is guaranteed by the following lemma.

Lemma 3. *There exists a function $f : \{0,1\}^n \to \{0,1\}^m$ satisfying the following two conditions:*

1. *for any advice* $\alpha \in \{0,1\}^{\bar{\ell}}$, $\Pr_{x \in \mathcal{U}_n}\left[R^{f,\bar{0};\alpha}(f(x)) \equiv x\right] \leq 2^{-\Omega(n)}$, *and*
2. *for any set* $S \subseteq \{0,1\}^{\bar{m}}$ *with* $|S| \leq 2^{(3/4)n}$, $\Pr_{\bar{x} \in \mathcal{U}_{\bar{n}}}\left[\bar{f}(\bar{x}) \in S\right] \leq \delta$.

Proof. First, note that giving R the oracle $\bar{0}$ does not help as any query to it can be answered by R itself without actually querying $\bar{0}$. Next, observe that for any f such that the corresponding function $\bar{f}$ does not satisfy the second condition, witnessed by the set S, the function C, defined as

$$C(\bar{y}) = \begin{cases} \text{any element from } \bar{f}^{-1}(\bar{y}) & \text{if } \bar{y} \in S, \\ 0^{\bar{n}} & \text{otherwise,} \end{cases}$$

can δ-invert $\bar{f}$ which implies that $R^{f,C;\alpha}$ can $(1-\varepsilon)$-invert f for some advice $\alpha \in \{0,1\}^{\bar{\ell}}$. Note that such a function C can be described by $|S|(\bar{m}+\bar{n})$ bits, so it can be replaced by an additional advice of that length. Thus, we can obtain from R another algorithm $\bar{R}$ which uses an advice of length at most $\bar{\ell} + |S|(\bar{m}+\bar{n}) \leq 2^{(4/5)n}$ such that if a function f fails on one of the conditions, then we have $\Pr_x[\bar{R}^{f;\bar{\alpha}}(f(x)) \equiv x] > 2^{-\Omega(n)}$ for some advice $\bar{\alpha}$. By Lemma 1, the fraction of such f's is less than one, which implies the existence of an f satisfying both conditions. □

Fix one such function f guaranteed by the lemma, and let $\bar{f} : \{0,1\}^{\bar{n}} \to \{0,1\}^{\bar{m}}$ be the corresponding harder function. To find an inverter for $\bar{f}$, we start from the function $\bar{f}^{-1}$, which clearly 1-inverts $\bar{f}$, and since it suffices to δ-invert $\bar{f}$, we can afford to destroy most of its outputs. More precisely, let $M : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$ be the *probabilistic* function (or equivalently, a distribution over deterministic functions) such that independently for any $\bar{y} \in \{0,1\}^{\bar{m}}$,

$$M(\bar{y}) = \begin{cases} \text{any element from } \bar{f}^{-1}(\bar{y}) & \text{with probability } 3\delta, \\ 0^{\bar{n}} & \text{with probability } 1-3\delta, \end{cases}$$

where we let $\bar{f}^{-1}(\bar{y}) = \{0^{\bar{n}}\}$ when $\bar{y} \notin \textsc{Image}(\bar{f})$. Then by a Markov inequality, we can have the following lemma showing that with a good probability, M inverts $\bar{f}$ well and thus can help R for inverting f.

Lemma 4. $\Pr_M\left[M \text{ } 2\delta\text{-}inverts \text{ } \bar{f}\right] \geq \delta$.

On the other hand, we would like to show that M is unlikely to help R for inverting f. More precisely, we would like to show that for any advice $\alpha \in \{0,1\}^{\bar{\ell}}$, the probability (over M) that $R^{f,M;\alpha}$ $(1-\varepsilon)$-inverts f is very small. The idea is that when R only makes a small number of queries, it has some chance of confusing the (useful) oracle M with the (useless) oracle $\bar{0}$.

Let us fix an advice $\alpha \in \{0,1\}^{\bar{\ell}}$ now. Consider the binary random variables V_x^α, for $x \in \{0,1\}^n$, defined as

- $V_x^\alpha = 1$ if and only if $R^{f,M;\alpha}(f(x)) \not\equiv x$.

Then we would like to give an upper bound on the probability

$$\Pr_M\left[\Pr_x\left[R^{f,M;\alpha}(f(x)) \not\equiv x\right] \leq \varepsilon\right] = \Pr_M\left[\sum_{x \in \{0,1\}^n} V_x^\alpha \leq \varepsilon 2^n\right].$$

However, a Markov inequality can only give an upper bound about $1 - 2\varepsilon$ (one can show that $\mathrm{E}[V_x^\alpha] \geq 3\varepsilon$ for most x), which is too large, while our goal is to have an upper bound of $2^{-\Omega(n)}$. For this we would like to apply Lemma 2.

However, there seem to be some obstacles preventing us from applying Lemma 2. First, many of these random variables may all depend on each other because the corresponding computations of R may all query M on some common entry. Second, even for a fixed x, the queries made by $R^{f,M;\alpha}(f(x))$ can vary for different M, as we allow R to make adaptive queries which can depend on the answers of previous queries.

To deal with the second obstacle, we consider another set of random variables Z_x^α, for $x \in \{0,1\}^n$, defined as

- $Z_x^\alpha = 1$ if and only if $M(\bar{y}) = 0^{\bar{n}}$ for any query $\bar{y}$ made by $R^{f,\bar{0};\alpha}(f(x))$.

Note that $R^{f,M;\alpha}(f(x)) = R^{f,\bar{0};\alpha}(f(x))$ if $Z_x^\alpha = 1$. Thus, for any x in the set $\textsc{Bad}^\alpha$, defined as

$$\textsc{Bad}^\alpha = \left\{ x \in \{0,1\}^n : R^{f,\bar{0};\alpha}(f(x)) \not\equiv x \right\},$$

we have $Z_x^\alpha \leq V_x^\alpha$, because $V_x^\alpha = 1$ if $Z_x^\alpha = 1$. Furthermore, by Lemma 3, $|\textsc{Bad}^\alpha| \geq 2^n(1 - 2^{-\Omega(n)}) \geq 2^n/2$.

Even when working with the variables Z_x^α's, we still face the first obstacle discussed above. To deal with this, we fix the values of M at those frequently queried entries. Call $\bar{y} \in \{0,1\}^{\bar{m}}$ *heavy* for an advice α if

$$\Pr_x \left[R^{f,\bar{0};\alpha}(f(x)) \text{ queries } \bar{0} \text{ at } \bar{y} \right] \geq w,$$

where we choose $w = 2^{-(2/3)n}$. Let $\hat{M}$ be the restriction of M defined as

$$\hat{M}(\bar{y}) = \begin{cases} 0^{\bar{n}} & \text{if } \bar{y} \text{ is heavy for some } \alpha, \\ M(\bar{y}) & \text{otherwise.} \end{cases}$$

Note that for each α, the number of heavy entries for α is at most q/w, because this number times w is at most the average number of queries made by $R^{f,\bar{0};\alpha}(f(x))$ over $x \in \{0,1\}^n$, which is at most q. Thus, the total number of all such heavy entries (over all α's) is at most $(q/w)2^{\bar{\ell}} \leq 2^{(3/4)n}$, since $2^{-cn} \leq \delta, \varepsilon$ and $\bar{\ell} \leq c\bar{n}$ for a small enough constant c. Let $\hat{V}_x^\alpha$ and $\hat{Z}_x^\alpha$ denote the random variables corresponding to V_x^α and Z_x^α, respectively, over the distribution of $\hat{M}$. Observe that now for each α, every variable $\hat{Z}_x^\alpha$ is mutually independent of all but $qw2^n$ other such variables, so it becomes possible to apply Lemma 2.

From now on, we will work with the distribution $\hat{M}$ and the random variables $\hat{V}_x^\alpha$ and $\hat{Z}_x^\alpha$, for $x \in \{0,1\}^n$. Let us see how this affects the arguments before when considering M, V_x^α, and Z_x^α. First, just as Lemma 4, we can show that such $\hat{M}$ also has a good chance of inverting $\bar{f}$ well.

Lemma 5. $\Pr_{\hat{M}}[\hat{M} \ \delta\text{-inverts } \bar{f}] \geq \delta$.

Proof. Let S be the set of heavy entries over all α's, which has $|S| \le 2^{(3/4)n}$, and we know that any $\bar{y}$ such that $\hat{M}(\bar{y}) \neq M(\bar{y})$ is contained in S. Since f satisfies the second condition of Lemma 3, we have

$$\Pr_{\bar{x}}\left[\hat{M}(\bar{f}(\bar{x})) \neq M(\bar{f}(\bar{x}))\right] \le \Pr_{\bar{x}}\left[\bar{f}(\bar{x}) \in S\right] \le \delta.$$

Thus, if M can 2δ-invert $\bar{f}$, $\hat{M}$ can δ-invert $\bar{f}$. Then from Lemma 4, we have the lemma. □

From this lemma and the guarantee of R, we have

$$\Pr_{\hat{M}}\left[\exists\alpha : R^{f,\hat{M};\alpha} \ (1-\varepsilon)\text{-inverts } f\right] \ge \delta$$

which implies the existence of an advice $\alpha \in \{0,1\}^{\bar{\ell}}$ such that

$$\Pr_{\hat{M}}\left[R^{f,\hat{M};\alpha} \ (1-\varepsilon)\text{-inverts } f\right] \ge \delta 2^{-\bar{\ell}}. \tag{1}$$

Let's fix one such α. On the other hand, $\Pr_{\hat{M}}[R^{f,\hat{M};\alpha} \ (1-\varepsilon)\text{-inverts } f]$ is

$$\Pr_{\hat{M}}\left[\sum_{x\in\{0,1\}^n} \hat{V}_x^\alpha \le \varepsilon 2^n\right] \le \Pr_{\hat{M}}\left[\sum_{x\in \textsc{Bad}^\alpha} \hat{V}_x^\alpha \le \varepsilon 2^n\right],$$

and since we still have $\hat{Z}_x^\alpha \le \hat{V}_x^\alpha$ for any $x \in \textsc{Bad}^\alpha$, the above is at most

$$\Pr_{\hat{M}}\left[\sum_{x\in \textsc{Bad}^\alpha} \hat{Z}_x^\alpha \le \varepsilon 2^n\right] \le \Pr_{\hat{M}}\left[\sum_{x\in \textsc{Bad}^\alpha} \hat{Z}_x^\alpha \le 2\varepsilon\,|\textsc{Bad}^\alpha|\right]$$

as $|\textsc{Bad}^\alpha| \ge 2^n/2$. Then we bound the last probability by the following.

Lemma 6. $\Pr_{\hat{M}}[\sum_{x\in\textsc{Bad}^\alpha} \hat{Z}_x^\alpha \le 2\varepsilon|\textsc{Bad}^\alpha|] < \delta 2^{-\bar{\ell}}$.

Proof. Note that for any $x \in \{0,1\}^n$,

$$\Pr_{\hat{M}}\left[\hat{Z}_x^\alpha = 1\right] \ge (1-3\delta)^q \ge 3\varepsilon,$$

as we assume $\varepsilon, \delta \le c$ and $q \le (c_0/\delta)\log(1/\varepsilon)$, for small enough constants c, c_0. By Lemma 2 with $k = |\textsc{Bad}^\alpha| \ge 2^n/2$, $A = \varepsilon k$, $d = 1 + qw2^n \le 2^{n/2}$, and $t = \varepsilon^2 2^{n/2}/16$, we have

$$\Pr_{\hat{M}}\left[\sum_{x\in \textsc{Bad}^\alpha} \hat{Z}_x^\alpha \le 2\varepsilon\,|\textsc{Bad}^\alpha|\right] \le 2\left(\frac{\varepsilon^2 2^n}{4\varepsilon^2 k}\right)^{t/2} \le 2\left(\frac{1}{2}\right)^{t/2} < \delta 2^{-\bar{\ell}},$$

since $2^{-cn} \le \delta, \varepsilon$ and $\bar{\ell} \le cn$ for a small enough constant c. (Note that the bound still holds even if we allow $\bar{\ell} \le 2^{cn}$.) □

This leads to a contradiction to the bound in (1). Therefore, the assumption we made at the beginning cannot hold, and R must make $\Omega((1/\delta)\log(1/\varepsilon))$ oracle queries, which proves Theorem 1.

3.2 Proof of Theorem 2

Note that what is different from Theorem 1 is that now we require that R makes only *non-adaptive* queries and as a result we allow R a much longer advice. Again, assume that R makes at most $q \leq (c_0/\delta)\log(1/\varepsilon)$ oracle queries, for a small enough constant $c_0 > 0$, and we will show that this leads to a contradiction.

Observe that in the proof of Theorem 1, why we can only have $\bar{\ell} \leq O(\bar{n})$ is that the restriction $\hat{M}$ fixes all the heavy entries over all α's at once, but there are $(q/w)2^{\bar{\ell}}$ such entries which can not exceed $2^{\bar{n}}$. Here, instead, we will consider different restrictions for different α's separately. Call $\bar{y} \in \{0,1\}^{\bar{m}}$ heavy for an advice α if

$$\Pr_x \left[R^{f,M;\alpha}(f(x)) \text{ queries } M \text{ at } \bar{y}\right] \geq w,$$

where $w = 2^{-(2/3)n}$, and note that this definition actually is independent of the choice of f and M as we assume that R makes only non-adaptive queries. As in the proof of Theorem 1, one can show that the number of heavy entries for any advice α is at most q/w. We will consider restricting an oracle function M (or $\bar{0}$) by fixing its values on those heavy entries for some α according to some function $\rho : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$. For an advice α and a function $\rho : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$, let M^{α}_{ρ} denote such a restriction of M, defined as $M^{\alpha}_{\rho}(\bar{y}) = \rho(\bar{y})$ if $\bar{y}$ is heavy for α and $M^{\alpha}_{\rho}(\bar{y}) = M(\bar{y})$ otherwise. Similarly, let $\bar{0}^{\alpha}_{\rho}$ denote such a restriction of $\bar{0}$. As in Lemma 3, we have the following lemma. Due to the space limitation, we omit its proof (which follows from Lemma 1, as $\bar{0}^{\alpha}_{\rho}$ has a short description and can be replaced by a short additional advice).

Lemma 7. *There exists a function $f : \{0,1\}^n \to \{0,1\}^m$ such that for any advice $\alpha \in \{0,1\}^{\ell}$ and any function $\rho : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$,*

$$\Pr_x \left[R^{f,\bar{0}^{\alpha}_{\rho};\alpha}(f(x)) \equiv x\right] \leq 2^{-\Omega(n)}.$$

Let us pick one such function f guaranteed by the lemma, let $\bar{f}$ be the corresponding harder function, and let M be the probabilistic function defined according to this $\bar{f}$. As in Lemma 4, we have $\Pr_M \left[M \text{ } \delta\text{-inverts } \bar{f}\right] \geq \delta$, and as before, this implies the existence of an advice $\alpha \in \{0,1\}^{\ell}$ such that $\Pr_M \left[R^{f,M;\alpha} \text{ } (1-\varepsilon)\text{-inverts } f\right] \geq \delta 2^{-\ell}$. Let us fix one such advice α. By an average argument, there must exist a restriction M^{α}_{ρ} of M which fixes the values of those heavy entries for α, such that

$$\Pr_{M^{\alpha}_{\rho}} \left[R^{f,M^{\alpha}_{\rho};\alpha} \text{ } (1-\varepsilon)\text{-inverts } f\right] \geq \delta 2^{-\ell}. \tag{2}$$

On the other hand, we will show a contradiction to this bound. Consider the set

$$\textsc{Bad}^{\alpha} = \left\{x \in \{0,1\}^n : R^{f,\bar{0}^{\alpha}_{\rho};\alpha}(f(x)) \not\equiv x\right\},$$

which by Lemma 7 has $|\textsc{Bad}^{\alpha}| \geq 2^n/2$. Let $\hat{V}^{\alpha}_x$ denote the binary random variable, over the distribution of M^{α}_{ρ}, defined as

– $\hat{V}_x^\alpha = 1$ if and only if $R^{f,M_\rho^\alpha;\alpha}(f(x)) \not\equiv x$.

Note that each variable $\hat{V}_x^\alpha$ is mutually independent of all but $qw2^n$ other variables. Then one can again show that for any $x \in \text{BAD}^\alpha$, $\Pr_{M_\rho^\alpha}[\hat{V}_x^\alpha = 1] \geq (1-3\delta)^q \geq 3\varepsilon$, and $\Pr_{M_\rho^\alpha}\left[R^{f,M_\rho^\alpha;\alpha} \text{ } (1-\varepsilon)\text{-inverts } f\right]$ is

$$\Pr_{M_\rho^\alpha}\left[\sum_{x\in\{0,1\}^n} \hat{V}_x^\alpha \leq \varepsilon 2^n\right] \leq \Pr_{M_\rho^\alpha}\left[\sum_{x\in\text{BAD}^\alpha} \hat{V}_x^\alpha \leq 2\varepsilon\,|\text{BAD}^\alpha|\right] < \delta 2^{-\ell},$$

by Lemma 2. This contradicts the bound in (2). Thus, R must make at least $\Omega((1/\delta)\log(1/\varepsilon))$ queries, which proves Theorem 2.

4 Randomness from Hardness

In this section, we show that any algorithm R realizing a weakly-black-box transformation from hardness to randomness must make many queries, unless it can use a long advice string. Our first result, Theorem 3 below, shows such a query lower bound for any R which is allowed to use an advice of linear length and to make adaptive queries. We will give the proof in Subsection 4.1.

Theorem 3. *Suppose an algorithm R uses an advice of length $\bar{\ell}$ and realizes a weakly-black-box transformation from $((1-\delta), n, m)$-hardness to $(\varepsilon, \bar{n}, \bar{m})$-randomness, with $2^{-cn} \leq \varepsilon, \delta \leq c$ and $\bar{\ell} \leq cn$ for a small enough constant $c > 0$. Then R must make at least $\Omega(n/\varepsilon^2)$ oracle queries.*

Our second result, Theorem 4 below, shows a query lower bound for any R which is even allowed an advice of exponential length but can only make non-adaptive queries. We will give the proof in Subsection 4.2.

Theorem 4. *Suppose an algorithm R uses an advice of length ℓ and realizes a weakly-black-box transformation from $((1-\delta), n, m)$-hardness to $(\varepsilon, \bar{n}, \bar{m})$-randomness, with $2^{-cn} \leq \varepsilon, \delta \leq c$ and $\ell \leq 2^{c\bar{n}}$ for a small enough constant $c > 0$. Then R must make at least $\Omega(n/\varepsilon^2)$ oracle queries, if it only makes* non-adaptive *queries.*

4.1 Proof of Theorem 3

Consider any R which realizes such a weakly-black-box transformation. Assume that R makes at most $q \leq c_0 n/\varepsilon^2$ oracle queries, for a small enough constant $c_0 > 0$, and we will show that this leads to a contradiction. The basic idea is that when R makes only a small number of queries, it is easy to get confused between some useful oracle $D : \{0,1\}^{\bar{m}} \to \{0,1\}$ (which is correlated with f) and a useless one $B : \{0,1\}^{\bar{m}} \to \{0,1\}$ (which is independent of f). Here, we take B to be a random function.

First, we would like to pick a function f which is hard to invert by $R^{f,B;\alpha}$ for any α. The existence of such a function is guaranteed by the following lemma,

which follows from Lemma 1 by observing that the oracle B, which is independent of f, can be simulated by R itself without needing to query it.

Lemma 8. *There exists a function* $f : \{0,1\}^n \to \{0,1\}^m$ *such that for any advice* $\alpha \in \{0,1\}^{\bar{\ell}}$, $\Pr_{x,B}\left[R^{f,B;\alpha}(f(x)) \equiv x\right] \leq 2^{-\Omega(n)}$.

Fix one such function f guaranteed by the lemma, and let $g = g_f : \{0,1\}^{\bar{n}} \to \{0,1\}^{\bar{m}}$ be the resulting generator. To find a distinguisher for g, let us start from the characteristic function of $\textsc{Image}(g)$, denoted as T (i.e., $T(u) = 1$ if and only if $u \in \textsc{Image}(g)$), which clearly can $(1 - 2^{-(\bar{m}-\bar{n})})$-distinguish g, and since we only need to ε-distinguish g, we can afford to add some noise to T. More precisely, let $N : \{0,1\}^{\bar{m}} \to \{0,1\}$ be a noise function such that independently for any $u \in \{0,1\}^{\bar{m}}$,

$$N(u) = \begin{cases} 1 & \text{with probability } \frac{1-4\varepsilon}{2}, \\ 0 & \text{with probability } \frac{1+4\varepsilon}{2}, \end{cases}$$

and consider the *probabilistic* distinguisher (or equivalently a distribution over deterministic distinguishers) $D = T \oplus N$ such that for any $u \in \{0,1\}^{\bar{m}}$, $D(u) = T(u) \oplus N(u)$. The following shows that D has a good chance of distinguishing g well.

Lemma 9. $\Pr_D[D \text{ } \varepsilon\textit{-distinguishes } g] \geq \varepsilon$.

Proof. From the definition, $\Pr_{x,D}\left[D(g(x)) = 1\right] - \Pr_{u,D}\left[D(u) = 1\right]$ is

$$\frac{1+4\varepsilon}{2} - \left(2^{-(\bar{m}-\bar{n})} \cdot \frac{1+4\varepsilon}{2} + \left(1 - 2^{-(\bar{m}-\bar{n})}\right) \cdot \frac{1-4\varepsilon}{2}\right) \geq 2\varepsilon,$$

since $\bar{m} \geq \bar{n} + 1$ and hence $2^{-(\bar{m}-\bar{n})} \leq \frac{1}{2}$. Then by a Markov inequality, we have the lemma. □

As before, from this lemma and the guarantee of R, one can show the existence of an advice $\alpha \in \{0,1\}^{\bar{\ell}}$ such that

$$\Pr_D\left[R^{f,D;\alpha} \text{ } \delta\text{-inverts } f\right] \geq \varepsilon 2^{-\bar{\ell}}. \tag{3}$$

Let us fix one such advice α.

On the other hand, we will show that this bound cannot hold as D is unlikely to help R for inverting f. The idea is that when R only makes a small number of queries, it behaves similarly according to the two different oracles D and B. More precisely, we will show that for most input x,

$$\Pr_D\left[R^{f,D;\alpha}(f(x)) \equiv x\right] \text{ is small } \textit{if } \Pr_B\left[R^{f,B;\alpha}(f(x)) \equiv x\right] \text{ is small.}$$

Here we choose not to show that the two probabilities have a small difference, as it would then only give a much smaller query lower bound.

Let c_1 be a small enough constant, and consider the set

$$\textsc{Bad}^{\alpha} = \left\{ x \in \{0,1\}^n : \Pr_B \left[R^{f,B;\alpha}(f(x)) \equiv x \right] \leq 2^{-c_1 n} \right\}. \tag{4}$$

From Lemma 8 and a Markov inequality, we have $\Pr_x [x \notin \textsc{Bad}^{\alpha}] \leq 2^{-\Omega(n)}$. Furthermore, we have the following.

Lemma 10. *For any $x \in \textsc{Bad}^{\alpha}$, $\Pr_D[R^{f,D;\alpha}(f(x)) \equiv x] \leq 2^{-\Omega(n)}$.*

Proof. Fix any $x \in \textsc{Bad}^{\alpha}$. First, let us consider the computations of $R^{f,B;\alpha}(f(x))$ over all possible instances of the oracle B, which can be seen as a tree in a natural way as follows. Each internal node corresponds to a query $u \in \{0,1\}^{\bar{m}}$ to the oracle B, which has two edges coming out for the two possible answers of $B(u)$, and each leaf contains the output of $R^{f,B;\alpha}(f(x))$ following the corresponding path of computation. We can assume without loss of generality that R always makes exactly q queries to the oracle B (by making dummy queries if necessary), and it never makes the same query twice on any path of computation (by remembering all previous queries and their answers). The tree has exactly 2^q leaves, and the bound of (4) implies that at most $L = 2^{-c_1 n} \cdot 2^q$ leaves have $R^{f,B;\alpha}(f(x)) \equiv x$.

Now let us see what happens to the probability bound in (4) when we change the oracle from B to D. While over a random B, each leaf is reached with the same probability 2^{-q}, this no longer holds if now we measure the probability over the distribution of D. Note that each edge corresponds to the bit $D(u) = T(u) \oplus N(u)$, for some u, and since T is fixed (as we have fixed f and thus g), we can label that edge by the bit $N(u)$. Then a leaf on a path with i labels of 1 is now reached with probability $\left(\frac{1-4\varepsilon}{2}\right)^i \left(\frac{1+4\varepsilon}{2}\right)^{q-i}$, which increases as i decreases. Observe that the sequences of labels on the 2^q paths to the leaves are all different. Thus, the probability $\Pr_D \left[R^{f,D;\alpha}(f(x)) \equiv x \right]$ can be bounded from above by the sum of the L largest probabilities among the leaves, which is at most

$$\sum_{i=0}^{t} \binom{q}{i} \left(\frac{1-4\varepsilon}{2}\right)^i \left(\frac{1+4\varepsilon}{2}\right)^{q-i}, \tag{5}$$

for any t such that $\sum_{i=0}^{t} \binom{q}{i} \geq L$. We can take $t = (\frac{1-5\varepsilon}{2})q$, which gives

$$\sum_{i=0}^{t} \binom{q}{i} \geq 2^{-O(\varepsilon^2 q)} \cdot 2^q \geq 2^{-c_1 n} \cdot 2^q \geq L,$$

since $q \leq c_0 n/\varepsilon^2$ and we can take c_0 to be much smaller than c_1. Then the bound in (5) is at most $2^{-\Omega(\varepsilon^2 q)} \leq 2^{-\Omega(n)}$ (using, for example, a Chernoff bound), which proves the lemma. □

Now let V_x^{α}, for $x \in \{0,1\}^n$, denote the binary random variable, over D, such that

- $V_x^{\alpha} = 1$ if and only if $R^{f,D;\alpha}(f(x)) \equiv x$.

We know from Lemma 10 that for any $x \in \text{BAD}^\alpha$, $\Pr_D[V_x^\alpha = 1] \leq 2^{-\Omega(n)}$. Then we have

$$\mathop{\mathrm{E}}_{D}\left[\sum_{x\in\{0,1\}^n} V_x^\alpha\right] \leq \sum_{x\in\text{BAD}^\alpha} \mathop{\mathrm{E}}_{D}[V_x^\alpha] + \sum_{x\notin\text{BAD}^\alpha} 1 \leq 2^{-\Omega(n)}\cdot 2^n < \varepsilon 2^{-\bar{\ell}}\delta\cdot 2^n,$$

since we assume $2^{-cn} \leq \varepsilon, \delta$ and $\bar{\ell} \leq cn$ for a small enough constant c. By a Markov inequality, we have

$$\Pr_D\left[R^{f,D;\alpha} \text{ } \delta\text{-inverts } f\right] = \Pr_D\left[\sum_x V_x^\alpha \geq \delta 2^n\right] < \varepsilon 2^{-\bar{\ell}}, \tag{6}$$

which contradicts the bound in (3). This implies that R must make at least $\Omega(n/\varepsilon^2)$ oracle queries, which proves Theorem 3.

4.2 Proof of Theorem 4

Note that what is different from Theorem 3 is that now we require that R makes only *non-adaptive* queries and as a result we allow R a much longer advice. Again, assume that R makes at most $q \leq c_0 n/\varepsilon^2$ oracle queries, for a small enough constant $c_0 > 0$, and we will show that this leads to a contradiction. The proof here will follow closely that for Theorem 3, except that at the end we will manage to get a smaller bound than that in inequality (6), by applying Lemma 2 instead of a Markov inequality. The idea is similar to that in the proof of Theorem 2.

Let $w = 2^{-(2/3)n}$, call an entry $\bar{y} \in \{0,1\}^{\bar{m}}$ heavy for an advice α if $\Pr_x[R^{f,D;\alpha}(f(x))$ queries D at $\bar{y}] \geq w$, and again one can show that the number of heavy entries for any α is at most q/w. Note that this definition is independent of the choice of D and f as we assume that R only makes non-adaptive queries. As in the proof of Theorem 2, we will consider restricting the functions D and B by fixing their values on those heavy entries for some α according to some function $\rho : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{n}}$, and let D_ρ^α and B_ρ^α denote the corresponding restrictions, respectively. Then similar to Lemma 7 and Lemma 8, one can show the existence of a function $f : \{0,1\}^n \to \{0,1\}^m$ such that for any advice $\alpha \in \{0,1\}^\ell$ and any function $\rho : \{0,1\}^{\bar{m}} \to \{0,1\}$,

$$\Pr_{x,B_\rho^\alpha}\left[R^{f,B_\rho^\alpha;\alpha}(f(x)) \equiv x\right] \leq 2^{-\Omega(n)}. \tag{7}$$

Let f be such a function guaranteed above, let $g = g_f$ be the resulting generator, and let D be the probabilistic distinguisher defined according to this g. As in Lemma 9, we have $\Pr_D[D \text{ } \varepsilon\text{-distinguishes } g] \geq \varepsilon$. Then as in the proof of Theorem 2, one can show that this implies the existence of an advice α and a restriction D_ρ^α of D which fixes the values of those heavy entries for α, such that

$$\Pr_{D_\rho^\alpha}\left[R^{f,D_\rho^\alpha;\alpha} \text{ } \delta\text{-inverts } f\right] \geq \varepsilon 2^{-\ell}. \tag{8}$$

Let us fix one such α and ρ.

On the other hand, we will show that the bound above can not hold. Let c_1 be a small enough constant, and consider the set:

$$\textsc{Bad}^{\alpha} = \left\{ x \in \{0,1\}^n : \Pr_{B_\rho^\alpha} \left[R^{f,B_\rho^\alpha;\alpha}(f(x)) \equiv x \right] \leq 2^{-c_1 n} \right\}.$$

By the bound in (7) and a Markov inequality, we have $\Pr_x [x \in \textsc{Bad}^{\alpha}] \geq 1/2$. Let $\hat{V}_x^\alpha$ denote the binary random variable, over the distribution of D_ρ^α, defined as

- $\hat{V}_x^\alpha = 1$ if and only if $R^{f,D_\rho^\alpha;\alpha}(f(x)) \equiv x$.

Note that each variable $\hat{V}_x^\alpha$ is mutually independent of all but $qw2^n$ other variables. Then using an argument similar to that in the proof of Theorem 3, one can show that for any $x \in \textsc{Bad}^{\alpha}$, $\Pr_{D_\rho^\alpha}[\hat{V}_x^\alpha = 1] \leq 2^{-\Omega(n)} \leq \delta/3$, and by Lemma 2, $\Pr_{D_\rho^\alpha}[R^{f,D_\rho^\alpha;\alpha}$ δ-inverts $f]$ is

$$\Pr_{D_\rho^\alpha} \left[\sum_{x \in \{0,1\}^n} \hat{V}_x^\alpha \geq \delta 2^n \right] \leq \Pr_{D_\rho^\alpha} \left[\sum_{x \in \textsc{Bad}^{\alpha}} \hat{V}_x^\alpha \geq (\delta/2) \left| \textsc{Bad}^{\alpha} \right| \right] < \varepsilon 2^{-\ell},$$

which contradicts the bound in (8). As a result, R must make at least $\Omega(n/\varepsilon^2)$ queries, which proves Theorem 4.

References

1. Chang, Y.-C., Hsiao, C.-Y., Lu, C.-J.: The impossibility of basing one-way permutations on central cryptographic primitives. J. Cryptology 19(1), 97–114 (2006)
2. Di Crescenzo, G., Malkin, T.G., Ostrovsky, R.: Single database private information retrieval implies oblivious transfer. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 122–138. Springer, Heidelberg (2000)
3. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. SIAM J. Comput. 35(1), 217–246 (2005)
4. Gertner, Y., Malkin, T., Reingold, O.: On the impossibility of basing trapdoor functions on trapdoor predicates. In: Proc. IEEE FOCS 2001, pp. 126–135 (2001)
5. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: Proc. IEEE FOCS 2000, pp. 325–335 (2000)
6. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
7. Goldreich, O., Levin, L.: A hard-core predicate for all one-way functions. In: Proc. ACM STOC 1989, pp. 25–32 (1989)
8. Goldreich, O., Impagliazzo, R., Levin, L., Venkatesan, R., Zuckerman, D.: Security preserving amplification of hardness. In: Proc. IEEE FOCS 1990, pp. 318–326 (1990)
9. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In: Proc. IEEE FOCS 1986, pp. 174–187 (1986)

10. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (1999)
11. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography. In: Proc. IEEE FOCS 1989, pp. 230–235 (1989)
12. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proc. ACM STOC 1989, pp. 44–61 (1989)
13. Kilian, J.: Founding cryptography on oblivious transfer. In: Proc. ACM STOC 1998, pp. 20–31 (1998)
14. Kushilevitz, E., Ostrovsky, R.: One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 104–121. Springer, Heidelberg (2000)
15. Lin, H., Trevisan, L., Wee, H.M.: On hardness amplification of one-way functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 34–49. Springer, Heidelberg (2005)
16. Lu, C.-J.: On the complexity of parallel hardness amplification for one-way functions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 462–481. Springer, Heidelberg (2006)
17. Naor, M.: Bit commitment using pseudorandomness. J. Cryptology 4(2), 151–158 (1991)
18. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Proc. ACM STOC 1990, pp. 387–394 (1990)
19. Shaltiel, R., Viola, E.: Hardness amplification proofs require majority. In: Proc. ACM STOC 2008, pp. 589–598 (2008)
20. Viola, E.: On constructing parallel pseudorandom generators from one-way functions. In: Proc. IEEE CCC 2005, pp. 183–197 (2005)
21. Yao, A.C.-C.: Theory and applications of trapdoor functions. In: Proc. IEEE FOCS 1982, pp. 80–91 (1982)
22. Zimand, M.: Exposure-resilient extractors and the derandomization of probabilistic sublinear time. Computational Complexity 17(2), 220–253 (2008)

On Randomizing Hash Functions to Strengthen the Security of Digital Signatures*

Praveen Gauravaram** and Lars R. Knudsen

Department of Mathematics
Technical University of Denmark
Matematiktorvet, Building S303
DK-2800 Kgs. Lyngby
Denmark
P.Gauravaram@mat.dtu.dk, Lars.R.Knudsen@mat.dtu.dk

Abstract. Halevi and Krawczyk proposed a message randomization algorithm called RMX as a front-end tool to the hash-then-sign digital signature schemes such as DSS and RSA in order to free their reliance on the collision resistance property of the hash functions. They have shown that to forge a RMX-hash-then-sign signature scheme, one has to solve a cryptanalytical task which is related to finding second preimages for the hash function. In this article, we will show how to use Dean's method of finding expandable messages for finding a second preimage in the Merkle-Damgård hash function to existentially forge a signature scheme based on a t-bit RMX-hash function which uses the Davies-Meyer compression functions (e.g., MD4, MD5, SHA family) in $2^{t/2}$ chosen messages plus $2^{t/2+1}$ off-line operations of the compression function and similar amount of memory. This forgery attack also works on the signature schemes that use Davies-Meyer schemes and a variant of RMX published by NIST in its Draft Special Publication (SP) 800-106. We discuss some important applications of our attack.

Keywords: Digital signatures, Hash functions, Davies-Meyer, RMX.

1 Introduction

The collision attacks on the MD5 [36] and SHA-1 [30] hash functions in the recent years are one of the most vital contributions in the field of cryptology [40, 41]. Since then, there has been a reasonable amount of research showing how seriously these attacks (in particular on MD5) could undermine the security of the digital signatures [24, 6, 17, 39, 38] in which these hash functions are deployed.

For a long time, it has been suggested to randomize the messages using a fresh random value, also called salt, before they are hashed and signed, in order to free

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* The work in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

** Author and this research project has been supported by the Danish Research Council for Technology and Production Sciences grant number 274-08-0052.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 88–105, 2009.

the security of the digital signatures from depending on the collision resistance of the hash functions [14,13,1]. Message randomization before hashing would make an attacker to predict the random value in order to make use of collisions in a hash function to forge a digital signature. In consequence of that, the security of a digital signature would be forced to depend on a property weaker than the collision resistance of the hash function.

At Crypto 2006, Halevi and Krawczyk [19] proposed and analysed two simple message randomization transforms as the front-end tools for any hash-then-sign signature scheme which uses Merkle-Damgård hash functions [26,9]. They have shown that these message randomization techniques would base the security of the hash-then-sign signature schemes on the second preimage resistance of the hash function. They have noted that long message second preimage attack of Kelsey and Schneier [23] can be mounted on the hashes of the randomized messages to find second preimages to existentially forge the signatures of these hashes. Merkle-Damgård hash functions with these front-end tools were also considered as new modes of operation [19]. One of these transforms, called RMX, has the practical viability with the signature schemes such as RSA [37] and DSA [29] and was fully specified in [19, Appendix D], [20, 21]. We call a hash-then-sign signature scheme which uses RMX as the front-end tool, a RMX-hash-then-sign signature scheme.

A signer computes the signature of a message m using a RMX-hash-then-sign signature scheme as follows: He chooses a random value denoted r, and randomizes m by passing the pair (r, m) as input to the RMX transform. The randomized message is given by $M = \mathrm{RMX}(r, m)$. The signer processes the message M using a t-bit hash function H and obtains the t-bit hash value $H(M)$. The signer signs the hash value $H(M)$ using a signature algorithm, denoted SIG, and obtains the signature s. The signer sends the triplet (m, r, s) to the verifier who computes $M = \mathrm{RMX}(r, m)$ and provides the pair (M, s) to the verification procedure to verify s.

The RMX transform requires no change to the signatures algorithms such as RSA and DSA. The implementation of RSA [37] based on RMX-SHA-1 was discussed in [21]. In 2007, NIST has published a variant of RMX as a draft special publication (SP) 800-106 [10] which was superseded in 2008 by a second draft SP 800-106 [11]. In addition, NIST intends that the candidate hash functions in the SHA-3 hash function competition support randomized hashing [31].

1.1 Related Work

Dang and Perlner [12] have shown a generic chosen message forgery attack on the signature schemes based on t-bit RMX-hashes in $2^{t/2}$ chosen messages and $2^{t/2}$ operations of the hash function and similar memory. This attack produces a collision of form $H(\mathrm{RMX}(r, m)) = H(\mathrm{RMX}(r^*, n))$ which implies $\mathsf{SIG}(m) = \mathsf{SIG}(n)$ where $(r, m) \neq (r^*, n)$, m is one of the chosen messages and n is the forgery message of m.

1.2 Our Results

In this article, we first note that the attack of Dang and Perlner [12] does not produce a verifiable forgery if the signer uses the same random value for both RMX hashing and signing such as in DSA [29, 32], ECDSA [3, 32] and RSA-PSS [37]. The re-use of the random value in the signature schemes for the randomized hashing to save the communication bandwidth was addressed in [19, 33, 27, 11, 12]. The attack of [12] also does not work on the signature schemes based on the other randomized hash function analysed by Halevi and Krawczyk [19] wherein both the salt and randomized hash value are signed to produce a signature.

We then show an existential forgery attack under a generic chosen message attack [18] on the RMX-hash-then-sign signature schemes when the compression functions of the hash functions have fixed points. Our attack produces a valid forgery in the above applications wherein the forgery attack of Dang and Perlner does not succeed. Our attack uses Dean's trick of finding fixed-point expandable messages [15, 23] for finding second preimages in the hash functions that use fixed point compression functions. Many popular hash functions, that include, MD4 [35], MD5 [36], SHA family [30] and Tiger [2] have compression functions that use Davies-Meyer construction [22, 34] for which fixed points can be easily found [28]. In an existential forgery attack, the attacker asks the signer for the signatures on a set of messages of his choice and is then able to produce a valid signature on a message which was never signed by the signer. Our forgery attack requires $2^{t/2}$ equal length chosen messages, $2^{t/2+1}$ off-line operations of the compression function and a probability of 2^{-24} to hit the correct bits used to pad the message by the RMX transform. The attack requires about $2^{t/2}$ memory. With this computational work, we can establish a collision of the form $H(\mathrm{RMX}(r, m)) = H(\mathrm{RMX}(r, m\|n))$ where m is one of the chosen messages and $n \neq m$ is a randomized message block which gives a fixed point. This implies $\mathsf{SIG}(m) = \mathsf{SIG}(m\|n)$ and we show the message $m\|n$ as the forgery of m. Our attack also works on the signature schemes that use a variant of RMX published by NIST in its SP 800-106 [11] and in its earlier version [10].

1.3 Impact of Our Results

Our forgery attack on the RMX-hash-then-sign signature schemes is totally impractical for the reasonable hash value sizes of 256 bits. Moreover, our attack can not be parallelizable as it requires a real signer to sign a huge set of messages. Our analysis is in no contradiction to that of Halevi and Krawczyk [19]. Moreover, it complements their analysis by showing that RMX-hashes achieve an essential security improvement with respect to off-line birthday attacks in forcing these attacks to work on-line (e.g. requiring $2^{t/2}$ messages signed by the legitimate signer and similar amount of memory). Our analysis demonstrates that the security of RMX-hash-then-sign signature schemes based on the t-bit ideal fixed-point compression functions is equivalent to that of the t-bit standard keyed hash function HMAC [4] based on a t-bit ideal compression function. The attack of [12] has a similar impact, though its application is limited.

1.4 Guide to the Paper

In Section 2, we provide the notation and the background information necessary to understand the paper. In Section 3, we describe randomized hash functions of [19] and outline the RMX specification and its variant published by NIST [11]. In Section 4, we discuss the forgery attack of [12] on the RMX-hash-then-sign schemes and its limitations. In Section 5, we show how to apply Dean's fixed point expandable messages to forge hash-then-sign signatures. In Section 6, we describe our forgery attack on the signature schemes based on the RMX hash mode and its variants. In Section 7, we conclude the paper with some open questions.

2 Preliminaries

In this section, we define some notation and review some fundamentals of hash functions and digital signatures that will be used throughout the paper.

2.1 Notation

The symbol $\|$ represents the concatenation operation. The notation e^{α} represents the concatenation of e bit α times where e is either 0 or 1. For example, $1^4 = 1\|1\|1\|1$. If a is a positive integer, we represent by $a^{[\alpha]}$, the first (from left to right) α bits of a. For example, if $a = 1011011001$ then $a^{[4]} = 1011$. Similarly, if a^b is a positive integer, we represent by $(a^b)^{[\alpha]}$, the first α bits of a^b.

2.2 Merkle-Damgård Hash Functions

Let $H : \{0,1\}^* \rightarrow \{0,1\}^t$ be a Merkle-Damgård hash function based on the compression function $h : \{0,1\}^b \times \{0,1\}^t \rightarrow \{0,1\}^t$. An upper bound in bits (say 2^l) on the length of the message to be hashed is often specified for H. The message m to be processed using H is split into blocks $m_1, m_2, \ldots, m_{L-1}$ and m_L. Let $|m_i| = b$ for $i = 1$ to $L-1$ and q be the number of the message bits in the last block m_L where $q < b$. If $q \leq b - l - 1$ then the message m is padded with $1\|0^{b-l-q-1}\||m|$ where $|m|$ is the l-bit binary representation of the length of the message m. If $q > b - l - 1$ then the message m is padded with $1\|0^{b-q-1}$ and a separate b-bit block $0^{b-l}\|\ |m|$ is concatenated to the padded message $m\|1\|0^{b-q-1}$. Every message block m_i, for $i = 1 \ldots L$, is processed using h as defined by $H_i = h(H_{i-1}, m_i)$ where H_0 is the initial value (IV) of H, H_i is the intermediate hash value of H at iteration i of h and H_L is the hash value of H. We denote by H^{H_0}, a hash function with H_0 as the IV.

Some properties of an ideal hash function H^{H_0}.

1. Collision resistance (CR): It should take about $2^{t/2}$ operations of H^{H_0} to find two messages m and n such that $m \neq n$ and $H^{H_0}(m) = H^{H_0}(n)$.
2. Second preimage resistance (SPR): For a challenged target message m, it should take about 2^t operations of H^{H_0} to find another message n such that $n \neq m$ and $H^{H_0}(m) = H^{H_0}(n)$. However, for a target message of 2^d blocks, second preimages for H^{H_0} can be found in about 2^{t-d} operations of h [23].

Some properties of an ideal compression function h.

1. CR: It should take about $2^{t/2}$ operations of h to find two different pairs (H_{i-1}, m_i) and (H^*_{i-1}, n_i) such that $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$.
2. SPR: For a challenged pair (H_{i-1}, m_i), it should take about 2^t operations of h to find a pair (H^*_{i-1}, n_i) such that $(H_{i-1}, m_i) \neq (H^*_{i-1}, n_i)$ and $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$. This property is also called random-SPR (r-SPR) [19].

2.3 Compression Functions with Fixed Points

A fixed point for a compression function h is a pair (H_{i-1}, m_i) such that $h(H_{i-1}, m_i) = H_{i-1}$. Let h be the Davies-Meyer compression function which is defined by $h(H_{i-1}, m_i) = E_{m_i}(H_{i-1}) \oplus H_{i-1} = H_i$ where m_i is the message block which is used as a key to the block cipher E and the input state H_{i-1} is the plaintext to E. A fixed point for h can be easily found by evaluating the expression $E^{-1}_{m_i}(0)$ for some message block m_i. Davies-Meyer feed-forward can also use addition mod 2^t and fixed points can still be found for this case. This technique to find fixed points for the Davies-Meyer construction has been described in [28], [23, Appendix A.1].

2.4 Existential Forgery Attack on the Signature Schemes

An existential forgery of a signature scheme SIG under a generic chosen message attack [18] (also called weak chosen message attack [8]) is performed as follows:

1. The attacker sends to the challenger (signer) a list of q messages $m^1, \ldots, m^q$.
2. The challenger generates a public and private key pair (Pk, Sk) using a key generation algorithm. The challenger generates the signatures s_i on the messages m^i computed using his private key Sk and the signature algorithm SIG for $i = 1, \ldots, q$. The challenger sends to the attacker the public key Pk and the signatures s_i.
3. The attacker forges the signature scheme SIG by outputting a pair (m, s) if:
   (a) $(m, s) \notin \{(m^1, s_1), \ldots, (m^q, s_q)\}$; and
   (b) The triplet (Pk, m, s) produces a valid verification.

Let Adv be the probability that the adversary wins the above game, taken over the coin tosses made by him and the challenger. The adversary is said to (t, q, ϵ)-existentially forge the signature scheme SIG if he runs in time at most t, makes at most q queries and $\mathsf{Adv} \geq \epsilon$.

3 Randomized Hashing

A family of hash functions $\{H_r\}_{r \in R}$ for some set R is target collision resistant (TCR) [5, 19] if no efficient attacker after choosing a message m and receiving the salt $r \in_R R$ can find a second preimage n such that $m \neq n$ and $H_r(m) = H_r(n)$

except with insignificant probability. The usage of the family $\{H_r\}_{r\in R}$ for the digital signatures necessitates the users to sign the salt r along with the hash of the message. However, hash-then-sign signature schemes such as DSA [29] and RSA [37] do not support signing the salt in addition to $H_r(m)$. In order to free such signature schemes from signing the salt, Halevi and Krawczyk introduced the notion of enhanced TCR (eTCR) hash function family [19]. The hash function family $\{\tilde{H}_r\}_{r\in R}$ is eTCR if there exists no efficient attacker who after committing to a message m and receiving the salt r, can find a pair $(r^*, n) \neq (r, m)$ such that $\tilde{H}_r(m) = \tilde{H}_{r^*}(n)$ except with insignificant probability.

Halevi and Krawczyk [19] presented two randomized hash function modes for H. The first t-bit scheme, denoted H_r, XORs every block m_i of the message m with a b-bit random value r as shown below:

$$H_r^{H_0}(m) = H_r^{H_0}(m_1\| \ldots \|m_L) \stackrel{\text{def}}{=} H^{H_0}(m_1 \oplus r\|m_2 \oplus r\| \ldots \|m_L \oplus r).$$

The second t-bit scheme, denoted $\tilde{H}_r$, prepends r to $m_i \oplus r$ for $i = 1 \ldots, L$ as shown below:

$$\tilde{H}_r^{H_0}(m) = \tilde{H}_r^{H_0}(m_1\| \ldots \|m_L) \stackrel{\text{def}}{=} H^{H_0}(r\|m_1 \oplus r\|m_2 \oplus r\| \ldots \|m_L \oplus r).$$

The functions $\tilde{H}_r$ and H_r are eTCR and TCR respectively if the compression function h is either chosen-SPR (c-SPR) or Evaluated-SPR (e-SPR) [19]. These properties for the compression function h are defined below:

1. c-SPR: For a given message block m_i, find $(H_{i-1}, H^*_{i-1}, n_i)$ such that $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$.
2. e-SPR: Choose $u \geq 1$ values $\Delta_1, \ldots, \Delta_u$ each of length b bits. Receive a random value $r \in \{0,1\}^b$ and then define $m_i = r \oplus \Delta_u$ and $H_{i-1} = H^{H_0}(r \oplus \Delta_1\| \ldots \|r \oplus \Delta_{u-1})$. Find (H^*_{i-1}, n_i) such that $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$.

A generic birthday attack can be mounted on the c-SPR property of h and it does not work on the r-SPR and e-SPR properties of h [19]. The eTCR construction $\tilde{H}_r$ was proposed as the preferred hash function mode for use in digital signatures as it does not require explicit signing of the salt r and allows for better implementation flexibility. A concrete specification of $\tilde{H}_r$ called RMX and its usage with the digital signatures and its implementation details were discussed in [19, Appendix D] [20] and [21] respectively. RMX was also considered as a message randomization transform.

3.1 RMX Specification

The RMX scheme randomizes an input message m of at most $2^l - b$ bits using a random value r of length between 128 and b bits to an output message M. The RMX algorithm is defined below following [19, Appendix D], [20, 21]:

1. Three random values r_0, r_1 and r_2 are computed from r as follows:
   (a) $r_0 = r\|0^{b-|r|}$ such that $|r_0| = b$ bits.

   (b) $r_1 = \underbrace{r\|r\|\ldots\|r}_{b \text{ bits}}$ such that $|r_1| = b$ and the last repetition of r is truncated if needed.
   (c) $r_2 = r_1^{[b-l-8]}$ (The first $b-l-8$ bits of r_1).
2. Split the input message m into $L-1$ b-bit blocks $m_1, m_2 \ldots, m_{L-1}$ and a last block m_L of length b' where $1 \leq b' \leq b$.
3. Set $M_0 = r_0$.
4. For $i = 1$ to $L-1$:
   (a) $M_i = m_i \oplus r_1$.
5. Let $lpad$ (meaning last block pad) be a 16-bit string, representing the bit length b' of m_L in the big-endian notation. If $lpad_0$ and $lpad_1$ are the first and second bytes of $lpad$, respectively, and each of these bytes represents a number between 0 and 255, then $b' = 256 \times lpad_1 + lpad_0$.
   (a) If $b' \leq b-l-24$ then set $M_L^* = m_L\|0^k\|lpad$ where $k = b-b'-16-l$. Set $M_L = M_L^* \oplus r_2$.
   (b) If $b' > b-l-24$ then set $M_L^* = m_L\|0^{b-b'}$ and $M_{L+1}^* = 0^{b-l-24}\|lpad$. Set $M_L = M_L^* \oplus r_1$ and $M_{L+1} = M_{L+1}^* \oplus r_2$.
6. Output the randomized message $M = \text{RMX}(r, m) = M_0\|\ldots\|M_L$ in the case of (5a) and $M = \text{RMX}(r, m) = M_0\|\ldots\|M_L\|M_{L+1}$ in the case of (5b).

Remark 1. We note that when $b' \leq b-l-24$, the padding rule designed for RMX with $k = b-b'-16-l$ requires a separate block to accommodate the padding and length encoding (at least $l+1$) bits of the message M required by the hash function H^{H_0} as illustrated in Appendix A. In addition, when $b' \leq b-l-24$, with $k = b-b'-16-l$, $|M_L^*| = b-l$ and hence $|r_2| = b-l$ bits which means $r_2 = r_1^{[b-l]}$. For example, let $|m_L| = b' = b-l-24$ bits. Then $k = b-b'-l-16 = b-(b-l-24)-l-16 = 8$ bits. Now $M_L^* = m_L\|0^8\|lpad$ and $|M_L^*| = b-l-24+8+16 = b-l$ bits. Hence, r_2 should also be $b-l$ bits. Hence, in this paper, we set $r_2 = r_1^{[b-l]}$ for $b' \leq b-l-24$ bits.

If we set $k = b-b'-24-l$ for $b' \leq b-l-24$ bits then the padding and length encoding bits required by H^{H_0} can be accommodated in the last block of M as illustrated in Appendix A. When $k = b-b'-24-l$, with $b' \leq b-l-24$, $|M_L^*| = b-l-8$ bits and hence $|r_2| = b-l-8$ bits which means $r_2 = r_1^{[b-l-8]}$ which is the same as in RMX specification. For example, let $|m_L| = b' = b-l-24$ bits. Then $k = b-b'-l-24 = b-(b-l-24)-l-24 = 0$ bits. Now $M_L^* = m_L\|0^0\|lpad$ and $|M_L^*| = b-l-24+0+16 = b-l-8$ bits and hence $|r_2| = b-l-8$ bits and we can set $r_2 = r_1^{[b-l-8]}$.

A variant of RMX. NIST has published a variant of RMX in its SP 800-106 [11] replacing its previous draft SP 800-106 [10]. We call the latest variant of RMX in SP 800-106 [11] by RMX_{SP} whose specification is placed in Appendix B.

Remark 2. We note that RMX and RMX_{SP} differ in the padding rule defined for the messages. In addition, in RMX, the prepended random value r is extended to a block of b bits by padding it with 0 bits, whereas, in RMX_{SP}, it is directly concatenated with the XOR of the message blocks and random value.

4 Generic Forgery Attack on the RMX-Hash-Then-Sign Signature Schemes

Dang and Perlner [12] proposed an on-line birthday forgery attack on the signature schemes based on t-bit RMX-hashes in $2^{t/2}$ chosen messages, $2^{t/2}$ off-line hash function operations and a similar amount of memory as outlined below:

- *On-line phase:*
  1. Query the signer for the signatures of $2^{t/2}$ chosen messages m^i where $i = 1, \ldots, 2^{t/2}$. Store every m^i in a Table L_1.
  2. The signer chooses a fresh random value r_i to compute the signature s_i of every message m^i using SIG. The signer first computes $\mathrm{RMX}(m^i)$ and then computes $\mathsf{SIG}(H^{H_0}(\mathrm{RMX}(m^i))) = s_i$. The signer returns the pair (r_i, s_i) where $i = 1, \ldots, 2^{t/2}$.
- *Off-line phase:*
  1. For $i = 1, \ldots, 2^{t/2}$, using r_i, compute the hash values $H^{H_0}(\mathrm{RMX}(r_i, m^i))$ and store them together with (r_i, s_i) in L_1.
  2. Choose random pairs (r_j, m_j) and compute the hash values $H^{H_0}(\mathrm{RMX}(r_j, m_j))$ where j is in increments of 1. While computing a hash value, check whether it collides with any of the hash values in the Table L_1. After about $j = 2^{t/2}$ attempts, with a good probability, we can find one collision. Let that random pair be (r_y, m_y). That is, we can find (r_x, m^x, H_x) from L_1 where $(r_x, m^x) \neq (r_y, m_y)$ and $H_x = H^{H_0}(\mathrm{RMX}(r_y, m_y)) = H^{H_0}(\mathrm{RMX}(r_x, m^x))$ where $x, y \in \{1, \ldots, 2^{t/2}\}$. Hence, $\mathsf{SIG}(m^x) = \mathsf{SIG}(m_y)$.
  3. Output message m_y as the forgery of m^x.

4.1 Limitations of the Forgery Attack

We note that the above attack does not produce a valid forgery in the following signature applications. These applications were noted in [19, 12].

- The random component that already exists in the signature schemes such as RSA-PSS, DSA and ECDSA, can also be used for randomized hashing (e.g., RMX hash mode) to save the bandwidth. The above attack does not succeed on such signature schemes during the forgery verification as the random value used by the signer for RMX hashing and signing matches the arbitrary value chosen by the attacker with a negligible probability.
- When the signature schemes based on TCR hashing H_r are used to sign a message m, both r and $H_r(m)$ have to be signed. For a valid forgery on such signatures, the attacker should use the same random value as the signer; hence this attack does not succeed.

5 Application of Dean's Fixed Point Expandable Messages to Forge Hash-Then-Sign Signature Schemes

Dean [15,23] has shown that if it is easy to find fixed points for the compression function then a fixed point expandable message, a multicollision using different length messages, can be constructed for the hash function. Using Dean's trick, we show that, one out of $2^{t/2}$ signatures obtained on the equal length messages from a legitimate signer can be forged by finding a collision which looks like $H^{H_0}(m) = H^{H_0}(m\|n)$ where n is the fixed point message block. This implies $\mathsf{SIG}(m\|n) = \mathsf{SIG}(m)$ and we show the message $m\|n$ as the forgery of m in $2^{t/2+1}$ invocations of H^{H_0} and one chosen message query to the signer. We assume that h is the Davies-Meyer compression function. The attack is outlined below:

1. Consider $2^{t/2}$ equal-length messages m^i of length $(c \times b) - (l+1)$ bits where c is the number of b-bit blocks and $i = 1, \ldots, 2^{t/2}$. Compute the hash values H_i of m^i under H^{H_0} where each m^i is padded with a bit 1 followed by l bits that represent the binary format of the length $(c \times b) - (l+1)$ bits. Let these $l+1$ bits be *pad* bits. Store m^i and H_i in a table L_1.
2. For $j = 1, \ldots, 2^{t/2}$, compute $2^{t/2}$ fixed points for h such that $h(H_j^*, n^j) = H_j^*$ where the last $l+1$ bits of every block n^j are fixed with a padding bit 1 and l bits that represent the binary format of the length of the message $m^i\|pad\|(n^j)^{[b-(l+1)]}$ where $(n^j)^{[b-(l+1)]}$ represents the first $b-(l+1)$ bits of n^j. Let the last $l+1$ bits of n^j be *padf* bits where by *padf* we mean padding bits in the fixed point block. Store H_j^* and $(n^j)^{[b-(l+1)]}$ in a table L_2.
3. According to the birthday paradox, with a significant probability, we can find a hash value H_x from the list L_1 and a hash value H_y^* from the list L_2 such that $H^{H_0}(m^x\|pad) = H_x = H_y^* = h(H_y^*, n^y)$ for some $x \in \{1, \ldots, 2^{t/2}\}$ and $y \in \{1, \ldots, 2^{t/2}\}$. This implies $H^{H_0}(m^x\|pad\|n^y) = H^{H_0}(m^x\|pad) = H_x$. Let $m = m^x$ and $n = (n^y)^{[b-(l+1)]}$.
4. Ask the signer for the signature on the message m. The signer hashes the message $m\|pad$ using H^{H_0} and then signs the hash value $H^{H_0}(m\|pad)$ using the signature algorithm SIG to obtain the signature $s = \mathsf{SIG}(m)$.
5. Now, $H^{H_0}(m\|pad\|n\|padf) = H^{H_0}(m\|pad)$. Hence, $\mathsf{SIG}(H^{H_0}(m\|pad)) = \mathsf{SIG}(H^{H_0}(m\|pad\|n))$ which implies $\mathsf{SIG}(m) = \mathsf{SIG}(m\|pad\|n)$. Note that *padf* bits are the padded bits to the message $m\|pad\|n$ when it is hashed with H^{H_0}.
6. Output the message $m\|pad\|n$ as the forgery of the chosen message m.

Remark 3. Our attack technique subtly differs from Dean's trick [15, 23] as we exert control over the fixed point message blocks by integrating the padding and length encoding bits that represent the length of the forgery message in the last few bits of the fixed point block. Whereas in Dean's trick to find expandable messages, all bits in the fixed point block can be random. Hence, our trick would also work to find expandable messages for the hash functions when the message inputs are XORed with a random value.

6 Existential Forgery Attack on Some RMX-Hash-Then-Sign Signatures

Here we extend the technique presented in Section 5 to provide an existential forgery attack on the hash-then-sign signatures that use t-bit fixed point compression functions and RMX transform specified in Section 3.1 and Remark 1. Our forgery attack also works in the applications outlined in Section 4.1 wherein the generic forgery attack described in Section 4 does not succeed.

In this attack, we first determine the length of the message to be forged and also length of the message to be produced as a forgery. We then compute $2^{t/2}$ fixed point message blocks for the compression function by integrating padding and length encoding bits required for the forgery message into the fixed point blocks. We then ask the signer to sign $2^{t/2}$ equal length chosen messages and collect their signatures and random values used for signing. We use those $2^{t/2}$ random values and messages to compute $2^{t/2}$ RMX-hashes and find a collision with the $2^{t/2}$ pre-computed fixed point hash values. We will find one of the chosen messages (along with its random value and signature) and one fixed point message block whose respective hashes collide. We then XOR the fixed point block and the random value to obtain a new block and concatenate it as a suffix block to the chosen message. Finally, we produce this concatenated message as the forgery of the chosen message. The attack involves some subtleties due to the fact that the RMX transform has a padding layer and hence, the attack has an additional negligible complexity of 2^{24} which adds to the complexity of $2^{t/2}$ chosen messages and $2^{t/2+1}$ operations of the compression function.

The pseudocode for the existential forgery attack on the signature scheme SIG which uses a hash function H^{H_0} based on the Davies-Meyer compression function h and RMX as the message randomization algorithm is given below:

- *Pre-computation phase:*
  1. Determine the length of the message to be forged. Let it be a message m of $2b-l-24$ bits. Let m^* be the forgery of m to be produced whose length can be pre-determined using $|m|$ as given by $|m^*| = |m|+l+24+b+(b-l-24-1) = 2b-l-24+l+24+b+b-l-25 = 4b-l-25$ bits.
  2. Pre-compute $2^{t/2}$ fixed points for the compression function h using message blocks N^j, each of size b bits, such that $H^*_{j-1} = h(H^*_{j-1}, N^j)$ for $j = 1, \ldots, 2^{t/2}$. While finding fixed points, fix the last $l+1$ bits of each message block N^j for the pad bit 1 and l bits to represent the pre-determined length encoding of the message m^* of length $4b-l-25$ bits to be produced as forgery. Let these $l+1$ bits be *padf* bits. Store the pairs (N^j, H^*_{j-1}) for $j = 1, \ldots, 2^{t/2}$ in a Table L_1.
- *On-line phase:*
  1. Query the signer with $2^{t/2}$ equal length chosen messages m^i for $i = 1, \ldots, 2^{t/2}$ (the message can also be the same in these queries). Let $|m^i| = b+b-l-24$ bits. Every message m^i can be represented as $m^i = m^i_1 \| m^i_2$ where $|m^i_1| = b$ bits and $|m^i_2| = b-l-24$ bits. Store these $2^{t/2}$ messages in a Table L_2.

For $i = 1, \ldots, 2^{t/2}$, the signer computes the signatures s_i on the equal-length messages m^i as follows:

(a) The signer chooses a fresh random value r_i for every query i independent of the message m^i. The signer calculates three random values $r_{0,i}$, $r_{1,i}$ and $r_{2,i}$ for every chosen random value r_i following the RMX specification in Section 3.1 and Remark 1 as follows:
   i. $r_{0,i} = r_i \| 0^{b-|r_i|}$
   ii. $r_{1,i} = r_i \| r_i \| \ldots \| r_i$ such that $|r_{1,i}| = b$ bits and the last repetition of r_i is truncated if needed.
   iii. $r_{2,i} = r_{1,i}^{[b-l]}$ (as noted in Remark 1).

(b) The signer splits every message m^i as $m^i = m_1^i \| m_2^i$ where $|m_1^i| = b$ bits and $|m_2^i| = b - l - 24$ bits.

(c) The signer randomizes every message m^i as follows:
   i. $M_0^i = r_{0,i}$
   ii. $M_1^i = m_1^i \oplus r_{1,i}$
   iii. $M_2^i = (m_2^i \| 0^8 \| lpad) \oplus r_{2,i}$

(d) Let $padm$ represents the l-bit binary encoded format of the length of the message $M_0^i \| M_1^i \| M_2^i$. For every randomized message $M_0^i \| M_1^i \| M_2^i$, the signer computes the hash value by passing this message as input to the hash function H^{H_0}. The hash value for every randomized message is $\tilde{H}_{r_i}^{H_0}(m^i) = H^{H_0}(M_0^i \| M_1^i \| (M_2^i \| 1 \| 0^{l-1}) \| (0^{b-l} \| padm))$. The signer returns the signature $s_i = \mathsf{SIG}(\tilde{H}_{r^i}^{H_0}(m^i))$ of every message m^i.

2. For every queried message m^i where $i = 1, \ldots, 2^{t/2}$, the signer returns the pair (r_i, s_i).

– *Offline phase:*

1. Add the received pair (r_i, s_i) to the table L_2.
2. Using the random value r_i, compute three random values $r_{0,i}$, $r_{1,i}$ and $r_{2,i}$ following the RMX specification in Section 3.1 and Remark 1.
3. Randomize the message m^i as follows:
   (a) $M_0^{i'} = r_{0,i}$
   (b) $M_1^{i'} = m_1^i \oplus r_{1,i}$
   (c) $M_2^{i'} = (m_2^i \| 0^8 \| lpad) \oplus r_{2,i}$
4. Let $M^{i'} = M_0^{i'} \| M_1^{i'} \| M_2^{i'}$. The validity of the signatures s_i returned by the signer during the *on-line* phase of the attack ensures that $M_0^{i'} \| M_1^{i'} \| M_2^{i'} = M_0^i \| M_1^i \| M_2^i$.
5. Compute $\tilde{H}_{r_i}^{H_0}(m^i) = H^{H_0}(M_0^i \| M_1^i \| (M_2^i \| 1 \| 0^{l-1}) \| (0^{b-l} \| padm))$ and add the hash values $\tilde{H}_{r_i}^{H_0}(m^i)$ to the Table L_2. Let $\tilde{H}_{r_i}^{H_0}(m^i) = H_i$ for $i = 1, \ldots, 2^{t/2}$. Now the Table L_2 contains the values (m^i, r_i, s_i, H_i).
6. Find a collision between the $2^{t/2}$ hash values stored in the Tables L_1 and L_2. With a significant probability, we can find a hash value H_x from the Table L_2 and a hash value H_y^* from the Table L_1 such that:

$$H^{H_0}(M_0^x \| M_1^x \| (M_2^x \| 1 \| 0^{l-1}) \| (0^{b-l} \| padm)) = H_x = h(H_y^*, N^y) = H_y^*$$

where
(a) $M_0^x = r_{0,x}$ and $|M_0^x| = b$
(b) $M_1^x = m_1^x \oplus r_{1,x}$ and $|M_1^x| = b$
(c) $M_2^x = (m_2^x \| 0^8 \| lpad) \oplus r_{2,x}$ and $|M_2^x| = b - l$
(d) $N^y = (N^y)^{[b-l-1]} \| padf$
and $x \in \{1, \ldots, 2^{t/2}\}$, $y \in \{0, \ldots, 2^{t/2} - 1\}$. This step is illustrated in Figure 1(a) and 1(b).

7. Let $r'_{2,x}$ be the last l bits of $r_{1,x}$. Then $r_{1,x} = r_{2,x} \| r'_{2,x}$. Note that $|r_{2,x}| = b-l$. Let $padr = r'_{2,x} \oplus (1 \| 0^{l-1})$. Let $padr1 = (r_{1,x}^{[b-l]} \oplus 0^{b-l}) \| (r'_{2,x} \oplus padm)$. Note that $|padr1| = b$ bits.
8. Calculate $(N^y)^{[b-l-1]} \oplus r_{1,x}^{[b-l-1]} = n^y$ as shown in Figure 1(c). The probability that the last 24 bits of n^y are $0^8 \| lpad$ is 2^{-24}. These 24 bits represent the padding bits of the message randomized by RMX.
9. Let $m^* = m_1^x \| (m_2^x \| 0^8 \| lpad \| padr) \| padr1 \| (n^y)^{[|n^y|-24]}$ and $m = m_1^x \| m_2^x$. Note that $|m| = 2b - l - 24$ bits and $m^* = 4b - l - 25$ bits which is the same as predetermined in Step (1) of the *pre-computation phase* of the attack. The signature $\mathsf{SIG}(m)$ on the message m is also valid on m^* as $\tilde{H}_r^{H_0}(m) = \tilde{H}_r^{H_0}(m^*)$.
10. Finally, output the message m^* as the forgery of the message m.

Complexity: It takes $2^{t/2}$ operations of h to *precompute* fixed points; $2^{t/2}$ chosen message queries to the signer during the *on-line phase*; $2^{t/2}$ operations of h and XOR operations during the *off-line phase* and a probability of 2^{-24} to hit the correct padding bits of the RMX transform. Total complexity of the attack is approximately $2^{t/2+1}$ operations of the compression function and $2^{t/2}$ chosen messages. The memory requirements of the attack are as follows assuming that the size of the signature is four times that of the security level of $t/2$ bits of a t-bit hash (which is possible in DSA): $b \times 2^{t/2} + t \times 2^{t/2}$ bits in the *precomputation phase*; $2b \times 2^{t/2}$ bits in the *on-line phase* and $|r| \times 2^{t/2} + t \times 2^{t/2} + 2t \times 2^{t/2}$ bits in the *off-line phase*. The total memory required for the attack is equal to $(3b + 4t + |r|) \times 2^{t/2}$ bits. This is approximately $2^{t/2+3}$ memory of t-bit values.

Illustration: Forging a signature scheme based on RMX-SHA-256 requires about 2^{129} operations of the SHA-256 compression function, 2^{128} chosen messages and a probability of 2^{-24}. Assuming that $|r| = 128$ bits, this attack requires a memory of about 2^{131}. In comparison, as noted in [19], forging a signature scheme based on RMX-SHA-256 using second preimage attack of [23] requires about 2^{201} SHA-256 compression function operations, more than 2^{55} memory and one chosen message.

Remark 4. Our forgery attack on the RMX-hash-then-sign signature schemes is independent of the size of the random value r. Our analysis assumes that in the RMX specification, $b' = b - l - 24$ bits. The attack also works for about the same complexity when $b' < b - l - 24$ bits. However, when $b' > b - l - 24$ bits, the padding bits of the RMX transform are placed in the last two blocks M_L and M_{L+1}. Since, the last block M_{L+1} does not contain any message bits, it is not possible to generate a fixed point block which can represent this block and hence, the attack does not work.

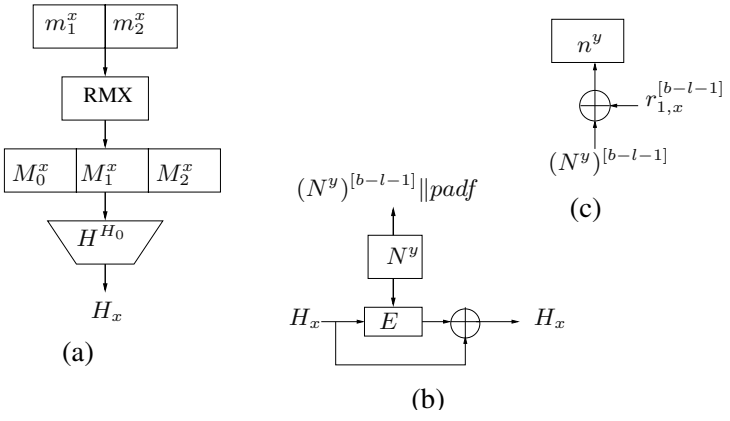


Fig. 1. Forgery attack on the RMX-hash-then-sign scheme based on Davies-Meyer

6.1 Applications of Our Forgery Attack

Our existential forgery attack on SIG based on RMX-hashes that use fixed point compression functions also works on SIG based on RMX_{SP}-hashes that use fixed point compression functions. When $|m| + 1 \geq |r|$ for RMX_{SP}, the complexity of our forgery attack on SIG using RMX_{SP} is similar to the one on SIG using RMX with the exception that it requires a success probability of 1/2 to hit the correct padding bit "1" used to pad the message by RMX_{SP}. The same attack also works on the signature schemes that use the previous version of RMX_{SP} [10] by assuming $|m| + 16 \geq |r|$. The attack has similar complexity as when RMX is used except that it has a success probability of 2^{-16} to hit the 16 padding bits used to pad the message by this variant.

Our forgery attack also works on the signatures based on the proposal $H^{H_0}(r\|H_r^{H_0}(m))$ [19] and on the signature schemes that use RMX transform together with the hash functions that use split padding [42] (assures that a minimum number of message bits are used in every block including the padding and length encoding block) to pad the message input to the hash function. Adding sequential counters to the RMX hash function (similar to the HAIFA hash mode [7]) also does not add any protection against our attack nor to the one in Section 4 as the counter inputs can be controlled in both the attacks. Note that sequential counters to the RMX hash function would still prevent the attempts to forge the RMX-hash-then-sign schemes using second preimage attacks of [23, 15].

Remark 5. Our on-line birthday forgery attack does not work on the signature schemes that use wide-pipe hash construction [25] with the internal state size $w \geq 2t$ based on the fixed point compression functions as the attack requires at least 2^t chosen messages and 2^{t+1} operations of the compression function. For example, Grøstl hash function [16], one of the selected candidates for the first round of NIST's SHA-3 hash function competition, uses a compression function for which fixed points can be easily found and has $w \geq 2t$ for a t-bit hash value.

6.2 Attack on the e-SPR Property of the Compression Functions

Our forgery attack on the RMX-hash-then-sign signature schemes translates into a birthday collision attack on the e-SPR property of the compression function h for which fixed points can be easily found. Recall that in the e-SPR game, we choose $u \geq 1$ values $\Delta_1, \ldots, \Delta_u$, each of length b bits. We then receive a random value $r \in \{0,1\}^b$ and define $m_i = r \oplus \Delta_u$ and $H_{i-1} = H^{H_0}(r \oplus \Delta_1 \| \ldots \| r \oplus \Delta_{u-1})$. Finally, we aim to find a pair (H^*_{i-1}, n_i) such that $(H^*_{i-1}, n_i) \neq (H_{i-1}, m_i)$ and $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$. The attack is outlined below:

1. Collect $2^{t/2}$ fixed point pairs (H_{i-1}, m^i) for h in a Table L where $i = 1, \ldots, 2^{t/2}$. Play the e-SPR game $2^{t/2}$ times always with $\Delta_1 = \Delta_2 = 0$ and every time we receive a fresh random value r_j for $j = 1, \ldots, 2^{t/2}$.
2. We check if $H_{i-1} = H^{H_0}(r_j \| r_j)$ for some i and j. Let that $r_j = r$ and fixed point be (H_{i-1}, m_i) where $m_i = m^i$ for some i.
3. Let $H^*_{i-1} = H^{H_0}(r)$, $n_i = r$, $H_{i-1} = H^{H_0}(r \| r)$. Now $h(H^*_{i-1}, n_i) = H^{H_0}(r \| r) = H^{H_0}(r \| r \| m_i) = h(H_{i-1}, m_i)$.

Thus, after an expected number of $2^{t/2}$ games, we win one game. Note that the forgery attack in Section 4 also translates into an e-SPR attack on any compression function after an expected number of $2^{t/2}$ e-SPR games.

7 Conclusion

Our research opens an interesting question on how to improve RMX SHA family without degrading its performance much to protect the signatures based on it against our forgery attack. One solution is not to mix the message bits processed using RMX transform with the padding and length encoding bits of the hash function by having only the latter bits in the last block. However, this patch introduces insufficient amount of randomness in the last block and requires some changes to the implementations of SHA family. It is an interesting problem to study this design. Recently, it has been suggested to use RMX-MD5 as a countermeasure to prevent impersonation attacks on the websites using collision attacks on MD5 [38]. Considering that one out of 2^{64} legitimate signatures based on RMX-MD5 can be forged following our results, it is an interesting problem to combine our techniques with those of [38] to analyse the security of RMX-MD5 over MD5 in the website certificates. Our research shows that randomized hashing is not easy to implement safely and we recommend NIST to consider our research during the SHA-3 hash function competition. It is clear from our attacks and those of [15,23] that it is well-worth investigating the SPR properties of the compression functions to identify possible weaknesses that may affect the randomized hashing setting.

Acknowledgments. Many thanks Quynh Dang and Ray Perlner for valuable discussions on this research and to Chris Mitchell for bringing to our awareness some important early works [14, 13, 1] on randomized hashing to strengthen the

digital signature security. Many thanks to our friends in the Information Security Group, RHUL for hosting Praveen Gauravaram to present preliminary results of this work and for their valuable discussions. Many thanks to the anonymous reviewers of Eurocrypt 2009 for some interesting comments and corrections; especially to the reviewer who suggested us to provide Section 6.2 and reminding us of the work of Dean [15]. Many thanks to our Crypto colleagues at MAT, DTU and Pierre-Alain Fouque for valuable discussions on this subject.

References

1. Akl, S.G.: On the Security of Compressed Encodings. In: Chaum, D. (ed.) Advances in Cryptology: Proceedings of Crypto 1993, pp. 209–230. Plenum Press, New York (1983)
2. Anderson, R., Biham, E.: Tiger: A Fast New Hash Function. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 89–97. Springer, Heidelberg (1996)
3. ANSI. ANSI X9.62:2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) (2005)
4. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
5. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making uOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
6. Bellovin, S., Rescorla, E.: Deploying a New Hash Algorithm. In: Proceedings of NDSS. Internet Society (Feburary 2006)
7. Biham, E., Dunkelman, O.: A Framework for Iterative Hash Functions - HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007) (Accessed on May 14, 2008), http://eprint.iacr.org/2007/278
8. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. Journal of Cryptology 21(2), 149–177 (2008)
9. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
10. Dang, Q.: NIST Special Publication 800-106 Draft Randomized Hashing Digital Signatures (2007) (Accessed on July 21, 2008), http://csrc.nist.gov/publications/drafts/Draft-SP-800-106/Draft-SP800-106.pdf
11. Dang, Q.: Draft NIST Special Publication 800-106 Draft Randomized Hashing Digital Signatures (2008) (Accessed on August 6, 2008), http://csrc.nist.gov/publications/drafts/800-106/2nd-Draft_SP800-106_July2008.pdf
12. Dang, Q., Perlner, R.: Personal communication (October 2008)
13. Davies, D., Price, W.: Security for Computer Networks. John Wiley, Chichester (1984)
14. Davies, D.W., Price, W.L.: The Application of Digital Signatures Based on Public-Key Cryptosystems. In: Proc. Fifth Intl. Computer Communications Conference, pp. 525–530 (October 1980)
15. Dean, R.D.: Formal Aspects of Mobile Code Security. PhD thesis, Princeton University (1999)
16. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl – A SHA-3 Candidate. First Round of NIST's SHA-3 Competition (2008) (Accessed on January 5, 2009), http://www.groestl.info/Groestl.pdf

17. Gauravaram, P., McCullagh, A., Dawson, E.: Collision Attacks on MD5 and SHA-1: Is this the "Sword of Damocles" for Electronic Commerce? In: Clark, A., McPherson, M., Mohay, G. (eds.) AusCERT Conference Refereed R & D Stream, pp. 1–13 (2006)
18. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
19. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006), http://www.ee.technion.ac.il/~hugo/rhash/rhash.pdf
20. Halevi, S., Krawczyk, H.: The RMX Transform and Digital Signatures (2006) (Accessed on July 30, 2008), http://www.ee.technion.ac.il/~hugo/rhash/rhash-nist.pdf
21. Halevi, S., Shao, W., Krawczyk, H., Boneh, D., McIntosh, M.: Implementing the Halevi-Krawczyk Randomized Hashing Scheme (2007) (Accessed on July 28, 2008), http://www.ee.technion.ac.il/~hugo/rhash/implementation.pdf
22. Hohl, W., Lai, X., Meier, T., Waldvogel, C.: Security of Iterated Hash Functions Based on Block Ciphers. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 379–390. Springer, Heidelberg (1994)
23. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
24. Lenstra, A.K., de Weger, B.: On the Possibility of Constructing Meaningful Hash Collisions for Public Keys. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 267–279. Springer, Heidelberg (2005)
25. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
26. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
27. Mironov, I.: Collision-Resistant No More: Hash-and-Sign Paradigm Revisited. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 140–156. Springer, Heidelberg (2006)
28. Miyaguchi, S., Ohta, K., Iwata, M.: Confirmation that Some Hash Functions Are Not Collision Free. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 326–343. Springer, Heidelberg (1991)
29. NIST. FIPS PUB 186-2: Digital Signature Standard (DSS) (January 2000) (Accessed on August 15, 2008), http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf
30. NIST. FIPS PUB 180-2-Secure Hash Standard (August 2002) (Accessed on May 18, 2008), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
31. NIST. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Docket No: 070911510-7512-01 (November 2007)
32. NIST. Draft FIPS PUB 186-3: Digital Signature Standard (2008) (Accessed on January 4, 2008), http://csrc.nist.gov/publications/drafts/fips_186-3/Draft_FIPS-186-3_November2008.pdf
33. Pasini, S., Vaudenay, S.: Hash-and-Sign with Weak Hashing Made Secure. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 338–354. Springer, Heidelberg (2007)

34. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
35. Rivest, R.L.: The MD4 Message Digest Algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
36. Rivest, R.: The MD5 Message-Digest Algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force (April 1992)
37. RSA Laboratories. *PKCS #1 v2.1: RSA Cryptography Standard*. RSA Data Security, Inc. (June 2002) (Accessed on August 15, 2008), `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`
38. Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., de Weger, B.: MD5 Considered Harmful Today Creating A Rogue CA Certificate. Presented at 25th Annual Chaos Communication Congress (2008) (Accessed on January 3, 2009), `http://www.win.tue.nl/hashclash/rogue-ca/`
39. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
40. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
41. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
42. Yasuda, K.: How to Fill Up Merkle-Damgård Hash Functions. In: Pieprzyk, J. (ed.) Advances in Cryptology - ASIACRYPT 2008. LNCS, vol. 5350, pp. 272–289. Springer, Heidelberg (2008)

A Observation in the Padding Rule of RMX

Consider hashing of a message m using RMX-SHA-256. For SHA-256, $b = 512$ bits. Let $|m| = 512 + 424 = 936$ bits where $|m_1| = 512$ and $|m_2| = 424$ bits. Following the specification of RMX given in Section 3.1, $b' = b - l - 24 = 512 - 64 - 24 = 424$ bits and $k = b - b' - 16 - l = 512 - 424 - 16 - 64 = 8$ bits. Let $|r| = 128$ bits. Now the randomized message M is defined as follows:

1. $M_0 = r_0$
2. $M_1 = m_1 \oplus r_1$
3. Calculation of M_2:
   (a) $M_2^* = m_2 \| 0^8 \| lpad$ where $|M_2^*| = 448$ bits
   (b) $M_2 = M_2^* \oplus r_2$

Therefore, $\text{RMX}(r, m) = M = M_0 \| M_1 \| M_2$. A hash function H^{H_0} used to process M requires at least $l+1$ bits for padding and length encoding. For SHA-256, $l = 64$ bits and hence it requires at least 65 bits for padding and length encoding. It is difficult to accommodate more than 64 bits in the remaining l-bit positions in the last block M_2 as it already has 448 bits. Therefore to process M using SHA-256, M is padded as follows: $M = M_0 \| M_1 \| \underbrace{(M_2 \| 1 \| 0^{63})}_{512 \text{ bits}} \| \underbrace{(0^{448} \| l)}_{512 \text{ bits}}$ where l represents the 64-bit binary encoded format of the length of M. Similarly, if

$b' = 423$ bits then $k = 9$ bits and $M_2^* = m_2 \| 0^9 \| lpad$. So, if $b' \leq b - l - 24$ then H^{H_0} requires an extra block to pad and length encode M.

Alternatively, when $b' \leq b - l - 24$ bits, we could define $k = b - b' - 24 - l$ bits. Then the hash function H^{H_0} does not require an extra block to length encode the message M. In the above illustration, when $|m| = 936$ bits, $M_2^* = m_2 \| 0^0 \| lpad$ and $M_2 = M_2^* \oplus r_2$ where $|M_2^*| = 440$ bits and $M = M_0 \| M_1 \| M_2$. To process M using a hash function H^{H_0}, M is padded as follows: $M = M_0 \| M_1 \| \underbrace{(M_2 \| 1 \| 0^7 \| l)}_{440+72 \text{ bits}}$.

B Message Randomization Technique RMX_{SP}

Let m be the input message, r be a message independent random bit string of at least 128 bits and at most 1024 bits and M be the randomized message. Let $zpad$ be a string of zero bits, which is zero or more "0" bits. Let λ denotes zero "0" bits or an empty string. Let $pad = 1 \| zpad$. Let $rpad$ be the 16-bit binary format of $|r|$. The input message m is encoded to the form $m \| pad$ and this encoded message is then randomized (transformed to M) as specified below.

1. If $|m| + 1 \geq |r|$:
   (a) $pad = 1 \| \lambda = 1$.
   Else
   (a) $pad = 1 \| 0^{|r| - |m| - 1}$.
2. $m' = m \| pad$.
3. If $|r| > 1024$ then stop and output an error indicator.
4. $rem = |m'| \bmod |r|$
5. Concatenate $\lfloor |m'|/|r| \rfloor$ copies of the r to the rem left-most bits of r to get R, such that $|R| = |m'|$. Now let

$$R = \underbrace{r \| r \| \ldots \| r}_{\lfloor |m'|/|r| \rfloor \text{ times}} \| r^{[rem]}$$

6. The randomized output is given by $M = \text{RMX}_{\text{SP}}(r, m) = r \| (m' \oplus R) \| rpad$.

Illustration: Let $|r| = 128$ and $|m| = 927$ bits. Now $|m| + 1 \geq r$, therefore $zpad = \lambda$ and $pad = 1$. Now $m' = m \| pad = m \| 1$ and $|m'| = 928$ bits. The random value $R = \underbrace{r \| \ldots \| r}_{7 \text{ times}} \| r^{[32]}$.

Cryptanalysis of MDC-2*

Lars R. Knudsen[1], Florian Mendel[2], Christian Rechberger[2], and Søren S. Thomsen[1]

[1] Department of Mathematics, Technical University of Denmark
Matematiktorvet 303S, DK-2800 Kgs. Lyngby, Denmark
[2] Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

Abstract. We provide a collision attack and preimage attacks on the MDC-2 construction, which is a method (dating back to 1988) of turning an n-bit block cipher into a $2n$-bit hash function. The collision attack is the first below the birthday bound to be described for MDC-2 and, with $n = 128$, it has complexity $2^{124.5}$, which is to be compared to the birthday attack having complexity 2^{128}. The preimage attacks constitute new time/memory trade-offs; the most efficient attack requires time and space about 2^n, which is to be compared to the previous best known preimage attack of Lai and Massey (Eurocrypt '92), having time complexity $2^{3n/2}$ and space complexity $2^{n/2}$, and to a brute force preimage attack having complexity 2^{2n}.

Keywords: MDC-2, hash function, collision, preimage.

1 Introduction

MDC-2 is a method of constructing hash functions from block ciphers, where the output size of the hash function is twice the size of the block cipher (hence it is called a *double-length construction*). MDC-2 was developed at IBM in the late 80s. A conference paper by IBM researchers Meyer and Schilling from 1988 describes the construction [21]. A patent was filed in August 1987, and the patent was issued in March 1990 [1]. The construction was standardised in ISO/IEC 10118-2 in 1994 [9]. It is mentioned in great detail in both the Handbook of Applied Cryptography [20, Alg. 9.46] and in the Encyclopedia of Cryptography and Security [27, pp. 379–380]. Furthermore, it is in practical use (see e.g., [10, 15, 26]).

Since publication, there seems to have been a wide belief in the cryptographic community that given an ideal block cipher, MDC-2 provides a collision resistant hash function. By this we mean that given an n-bit block cipher (thus yielding a $2n$-bit hash function), the required effort to find a collision in the hash function is expected to be 2^n. However, there is no proof of this property. The only proof

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* This work has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 106–120, 2009.

that collision resistance is better than $2^{n/2}$, as offered by many simpler (single-length) constructions, is due to Steinberger [25], who showed that for MDC-2 based on an ideal cipher, an adversary asking less than $2^{3n/5}$ queries has only a negligible chance of finding a collision.

In this paper we provide the first collision attack on MDC-2 which breaks the birthday bound. The attack makes no non-standard assumptions on the underlying block cipher. When applied to an instantiation of MDC-2 with e.g., a 128-bit block cipher (see e.g., [28]), the attack has complexity about $2^{124.5}$, which is better than the expected 2^{128} collision resistance for an ideal 256-bit hash function.

We also present improved preimage attacks on MDC-2. The previous best known preimage attack, first described by Lai and Massey [16], has time complexity about $2^{3n/2}$ and requires around $2^{n/2}$ memory. In this paper we provide a range of time/memory trade-offs, the fastest of which is significantly faster than the Lai/Massey attack. We describe attacks of any time complexity from 2^n to 2^{2n}. The memory requirements are such that the product of the time and space complexities is always around 2^{2n}. Hence, our most efficient preimage attack has time and space complexity about 2^n.

Finally, we describe how to use the preimage attack to find multicollisions faster than by the previous best known multicollision attack of Joux [11].

Related work. As mentioned, Lai and Massey described [16] a preimage attack on MDC-2 of complexity around $2^{3n/2}$. Knudsen and Preneel gave [14] a preimage attack on MDC-4 (a stronger and less efficient variant of MDC-2, to which the attacks described in this paper do not apply) of complexity $2^{7n/4}$. Steinberger proved [25] a lower bound of $2^{3n/5}$ for collision resistance of MDC-2 in the ideal cipher model.

Our attacks in fact apply to a larger class of hash function constructions based on block ciphers (see Section 2.1). Knudsen, Lai and Preneel described [13] collision and preimage attacks on all block cipher based hash function constructions of rate 1, meaning that one message block is processed per block cipher call. These attacks do not apply to MDC-2 (having rate 1/2).

Recently, a number of new double-length constructions have been proposed. At FSE 2005, Nandi et al. [23] proposed a rate 2/3 scheme, and they proved that finding a collision requires at least $2^{2n/3}$ queries. Later the same year (Indocrypt 2005), Nandi [22] introduced a class of rate 1/2 double-length schemes, all instances of which having optimal collision resistance 2^n. At Asiacrypt 2005, Lucks [18] proposed the double-pipe scheme as a failure-friendly design, meaning that collision resistance is retained even if the underlying compression function slightly fails to be collision resistant. The scheme maintains two chains, which are combined at the end, and hence is in fact a single-length scheme. However, by omitting the merging at the end one has a double-length scheme, which is optimally collision resistant. Hirose [8] proposed (FSE 2006) a collision resistant double-length scheme, based on an n-bit block cipher accepting keys of more than n bits. The rate depends on the key size. For all these schemes, the security proof is based on the assumption that the underlying primitive (compression function or block cipher) is secure. Our attacks do not apply to any of the schemes mentioned here.

Hellman has described a generic method to find a preimage of a $2n$-bit hash function with runtime $2^{4n/3}$ [7]. The caveat is that (apart from requiring $2^{4n/3}$ memory) a precomputation of cost 2^{2n} is needed. The preimage attacks on MDC-2 that are described in this paper are on a much better time/memory trade-off curve, and do not require a 2^{2n} precomputation.

2 Preliminaries

The collision attack presented in this paper makes use of *multicollisions*.

Definition 1. *Let f be some function. An r-collision for f is an r-set $\{x_1, \ldots, x_r\}$ such that $f(x_1) = \ldots = f(x_r)$. A multicollision is an r-collision for some $r > 1$. A 2-collision is known simply as a collision.*

Consider the classical occupancy problem (see e.g., [5]) consisting of randomly throwing q_1 balls into 2^n urns, where it is assumed that each of the 2^{nq_1} possible outcomes is equally likely. In order for the probability that at least one urn contains at least r balls to be $1 - 1/e$, one must throw about

$$q_1 = (r!2^{n(r-1)})^{1/r} \tag{1}$$

balls in total [5, IV,(2.12)]. The classical occupancy problem can be translated into the problem of finding an r-collision for a sufficiently random n-bit function f. Hence, this task has expected complexity q_1 as given by (1). In the following we shall use this expression as an estimate for the complexity of finding an r-collision.

We note that a standard birthday collision attack has complexity $2^{(n+1)/2}$, according to (1) with $r = 2$. With $2^{n/2}$ queries a collision is found with probability about $1 - e^{-1/2} \approx 0.39$.

2.1 Description of the MDC-2 Construction

MDC-2 was originally defined using DES [24] as the underlying block cipher. Here, we think of MDC-2 as a general double-length construction method for hash functions based on block ciphers. For ease of presentation we shall assume that keys and message blocks are of the same size, even if this is in fact not the case for DES. In Appendix A, we discuss this special case.

Let $E_K(m)$ denote the encryption under some block cipher (assumed to be secure) of plaintext m using the key K. If X is an n-bit string, then we let X^L denote the leftmost $n/2$ bits of X, and we let X^R denote the rightmost $n/2$ bits of X. Given E, MDC-2 defines a $2n$-bit hash function (with some given, distinct initial values H_0 and $\tilde{H}_0$) as follows. Split the message M (assumed to be appropriately padded) into t blocks $m_1, \ldots, m_t$, and do, for each i from 1 to t, the following ('$\|$' denotes concatenation).

$$V = E_{H_{i-1}}(m_i) \oplus m_i$$
$$\tilde{V} = E_{\tilde{H}_{i-1}}(m_i) \oplus m_i,$$

followed by

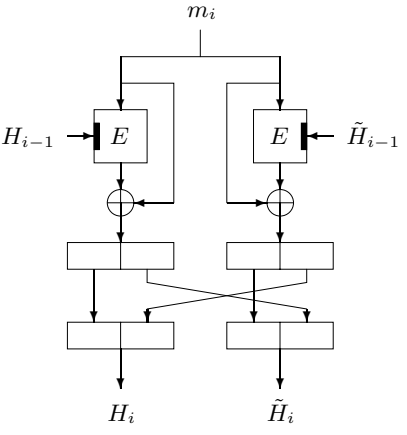


Fig. 1. The MDC-2 construction

$$H_i = V^L \| \tilde{V}^R$$
$$\tilde{H}_i = \tilde{V}^L \| V^R.$$

The output is $H_t \| \tilde{H}_t$. See also Figure 1. In other words, the chaining variables H_{i-1} and $\tilde{H}_{i-1}$ are used as keys in two block cipher calls, which each encrypt the message block m_i, and subsequently xor the resulting ciphertexts with m_i. The two right halves of the results are then swapped to obtain the next pair of chaining variables. In what follows, these steps will be called *an iteration*.

In the original description of MDC-2 [21], two bits of each of the two keys H_{i-1} and $\tilde{H}_{i-1}$ were fixed. This had two implications. First of all, all known weak and semi-weak keys of DES were ruled out, and secondly, this measure ensured that the two keys were always different. There seems to be no strong consensus that fixing key bits is a necessary security measure when MDC-2 is based on some other block cipher for which weak keys are not believed to exist. However, one might argue that ensuring that the two keys are different increases security – although this practice also has a cost in terms of security: the amount of state passed on from one iteration to the next is less than $2n$ bits. The attacks presented in this paper can be applied regardless of whether or not some key bits are fixed. However, the discussion of Section 6 assumes that no key bits are fixed.

A generalisation. We may generalise the MDC-2 construction. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be any function, and let g be any (efficiently invertible) bijection from $2n$ bits to $2n$ bits. Then a generalised construction is the following.

$$\begin{aligned} W &= f(H_{i-1}, m_i) \| f(\tilde{H}_{i-1}, m_i) \\ H_i \| \tilde{H}_i &= g(W). \end{aligned} \tag{2}$$

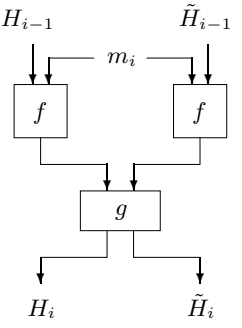


Fig. 2. The generalised MDC-2 construction

See Figure 2. In standard terms, (2) defines a *compression function* $h : \{0,1\}^{3n} \rightarrow \{0,1\}^{2n}$. The attacks presented in this paper apply to any instance of this construction. Notice that MDC-2 has $f(x,y) = E_x(y) \oplus y$ and $g(a\|b\|c\|d) = a\|d\|c\|b$. In the following we shall use the notation of the generalised construction. We assume that evaluating g (both forwards and backwards) costs much less than evaluating f. Our complexity estimates will be in terms of compression function evaluations. For example, if an attack requires T calls of f, we shall count this as having time complexity $T/2$, since f is evaluated twice in the compression function.

3 The Collision Attack

The collision attack applies to any construction of the type (2). We use the notation of Section 2 in the following description of the collision attack.

1. Given initial chaining values H_0 and $\tilde{H}_0$, find an r-collision in H_1. Let the messages producing the r-collision be $m_1^1, \ldots, m_1^r$, and let the r ("random") values of $\tilde{H}_1$ be $\tilde{H}_1^1, \ldots, \tilde{H}_1^r$.
2. Let $\ell = 1$.
3. Choose the message block m_2^ℓ arbitrarily, and evaluate $W_j^\ell = f(\tilde{H}_1^j, m_2^\ell)$ for every j, $1 \leq j \leq r$. If $W_i^\ell = W_j^\ell$ for some $i \neq j$, $1 \leq i, j \leq r$, then a collision $(m_1^i\|m_2^\ell, m_1^j\|m_2^\ell)$ has been found. If not, increment ℓ and repeat this step.

See Figure 3. Step 1 requires finding an r-collision in an n-bit function. This is expected to take time $q_1 = (r!2^{n(r-1)})^{1/r}$ as mentioned in Section 2. The probability of success in Step 3 is about $\binom{r}{2}2^{-n}$, since there are $\binom{r}{2}$ pairs of n-bit values, which may be equal. Hence, we expect to need to repeat Step 3 $2^n/\binom{r}{2}$ times. In each iteration we evaluate the encryption function r times. In the construction (2), f is evaluated twice per message block, and hence the r evaluations of f are equivalent to $r/2$ compression function evaluations. The total work required in Step 3 is therefore expected to be

$$q_2 = (r/2) \cdot 2^n / \binom{r}{2} = 2^n/(r-1).$$

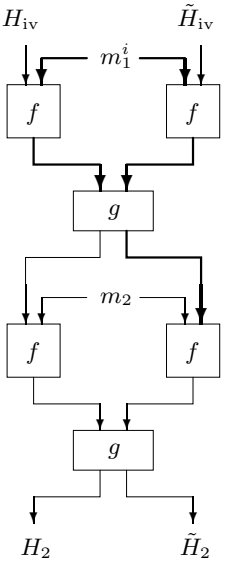


Fig. 3. The collision attack. Thick lines mean that there are r different values of this variable. Thin lines mean that there is only one.

Table 1. Time complexity of the collision attack on MDC-2 with an n-bit block cipher, compared to birthday complexity. For details in the case of MDC-2 based on DES ($n = 54$), see Appendix A.1.

n	r	Collision attack complexity	
		Section 3	Birthday
54	8	$2^{51.5}$	2^{54}
64	9	$2^{61.3}$	2^{64}
128	14	$2^{124.5}$	2^{128}
256	24	$2^{251.7}$	2^{256}

The total work required is $q_1 + q_2 = (r!2^{n(r-1)})^{1/r} + 2^n/(r-1)$. Hence, we may choose r as the integer ≥ 2 that minimises this expression. Notice that q_1 is an increasing function of r, and q_2 is decreasing. By setting $q_1 = q_2$ one gets, very roughly, a time complexity around $(\log_2(n)/n)2^n$. However, it turns out that the best choice of r is not exactly the one where $q_1 = q_2$, as one might expect. Table 1 shows the best choices of r and the corresponding complexities for different sizes n of the block cipher.

The probability of success of our attack with these complexities is about $1-1/e$ for Step 1, and the same probability for Step 3 when repeated $2^n/\binom{r}{2}$ times, in total $(1-1/e)^2 \approx 0.40$. As mentioned in Section 2, the probability of success for the birthday attack with 2^n queries is about $1 - e^{-1/2} \approx 0.39$. Hence, we consider the comparisons fair.

4 Preimage Attacks

A brute force preimage attack on MDC-2 (or on (2) in general) has time complexity $O(2^{2n})$ and space complexity $O(1)$. The previous best known preimage attack is due to Lai and Massey [16], and has time complexity $O(2^{3n/2})$ and space complexity $O(2^{n/2})$. Hence, for both attacks the product of the time complexity and the space complexity is $O(2^{2n})$. In the following subsection we describe a range of preimage attack time/memory trade-offs, for which the product of the time and the space complexities is at most $n2^{2n}$, but where time complexity can be anything between $O(n2^n)$ and $O(2^{2n})$. In Section 4.2 we describe how to reach a time and space complexity of $O(2^n)$.

4.1 An Attack Allowing for Time/Memory Trade-Offs

The attack uses pseudo-preimages, which are preimages of the compression function where both the chaining value and the message block can be chosen freely by the attacker. The attack can be outlined as follows.

1. Build a binary tree of pseudo-preimages with the target image $H_{\mathrm{T}}\|\tilde{H}_{\mathrm{T}}$ as root: the nodes are labelled with intermediate hash values, and each edge is labelled with a message block value meaning that this message block maps from the intermediate hash value at the child node to the intermediate hash value at the parent. The tree has (on average) two children for each node, and it has depth d meaning there are 2^d leaves.
2. From the initial value $H_{\mathrm{iv}}\|\tilde{H}_{\mathrm{iv}}$ of the hash function, find a message block that produces an intermediate hash value equal to one of the leaves in the tree from Step 1.

See Figure 4. The above technique clearly leads to a preimage consisting of a message block that maps to a leaf ℓ in the tree, and a sequence of d message blocks corresponding to the path in the tree that leads from the leaf ℓ to the root. Hence the total length of the message is $d + 1$ blocks.

The value of d determines the time/memory trade-off. We shall discuss concrete values of d later. The cost of Step 1 will be evaluated in the following.

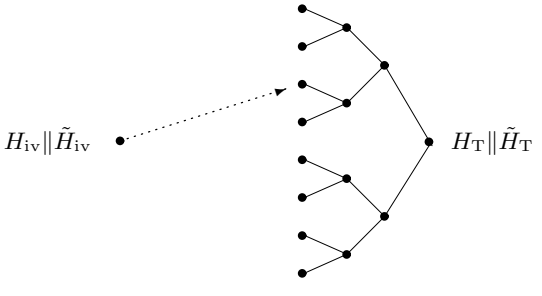


Fig. 4. A binary tree of pseudo-preimages of depth $d = 3$

Since the tree has 2^d leaves, Step 2 is expected to take time 2^{2n-d}. In effect, by constructing the tree we produce 2^d new target images, which improves the efficiency of the final brute force search by a factor of 2^d. The memory requirements are $2^d + 2^{d-1} + \ldots + 1 = 2^{d+1} - 1$ intermediate hash values.

We note that the last message block, the one that maps to the target image, must contain proper padding for a message of $d + 1$ blocks. If there are not enough degrees of freedom in the last block to both ensure proper padding and to find two pseudo-preimages, then a few initial steps (consisting of finding a small number of pseudo-preimages) are needed to ensure proper padding. It will become clear in the following that this only has a small effect on the total time complexity.

Constructing the tree (Step 1 above) is very time consuming for an ideal hash function. However, for the MDC-2 construction, there is an efficient method based on the following theorem.

Theorem 1. *Given a target hash value $H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$, a pseudo-preimage can be found in time at most 2^{n-1} with probability about $(1 - 1/e)^2$. By a pseudo-preimage we mean a pair $(H_{\mathrm{p}}, \tilde{H}_{\mathrm{p}})$ and a message block m such that $g(f(H_{\mathrm{p}}, m) \| f(\tilde{H}_{\mathrm{p}}, m)) = H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$.*

Proof. The method is the following. Let $U \| \tilde{U} = g^{-1}(H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}})$. Choose m arbitrarily, define $f_m(x) = f(x, m)$, and evaluate f_m on all $x \in \{0,1\}^n$. Referring again to the classical occupancy problem, when randomly throwing 2^n balls into 2^n urns, the probability that a given urn contains at least one ball is about $1-1/e$. Assuming that f_m is sufficiently random, this means that the probability that a given image has at least one preimage is about $1 - 1/e$, and additionally assuming independence, it means that the probability of finding at least one preimage of both U and $\tilde{U}$ is $(1 - 1/e)^2$. Let these preimages be H_{p} and $\tilde{H}_{\mathrm{p}}$, respectively. Then $g(f_m(H_{\mathrm{p}}) \| f_m(\tilde{H}_{\mathrm{p}})) = H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$. Finally, the complexity of evaluating f_m 2^n times corresponds to 2^{n-1} compression function evaluations. □

We note that for an ideal $2n$-bit compression function, the above task has complexity about 2^{2n}. The story does not finish with Theorem 1, however. Clearly, by evaluating a random n-bit function 2^n times, one finds on average one preimage for all elements of $\{0,1\}^n$. Thus, we obtain the following corollary.

Corollary 1. *Given t target hash values, in time 2^{n-1} one pseudo-preimage (on average) can be found for each target hash value. Here, t can be any number between 1 and 2^n.*

Proof. The technique is the same as above (we note that inverting g, which must be done 2^t times, is assumed to be a much simpler task than evaluating f). Since f_m is evaluated on all 2^n possible inputs, on average one preimage is found for each element of $\{0,1\}^n$. Therefore, again assuming independence, we also expect one preimage on average of each of the t target hash values. With respect to the complexity, we repeat that 2^n calls to f_m is equivalent to about 2^{n-1} compression function calls. □

In the case of MDC-2, where g has a special form that allows to compute n bits of the output given only n bits of the input (and vice versa), t above can actually be 2^{2n} without affecting the complexity. The reason is that g (in this case) never has to be inverted more than 2^n times.

Due to Theorem 1 and Corollary 1, the tree described above can be efficiently constructed as follows (note that the tree will, in fact, not be binary, due to some nodes having no children, and others having more than two, but on average the number of children per node will be two):

Assign the value $H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$ of the target image to the root of the tree. Then find (in expected time 2^n) two pseudo-preimages of the target image by the method of Theorem 1 (applied twice with different message blocks m). This means the tree now contains the root and two children of the root. Then find two pseudo-preimages of each of the two children of the root. This also takes time 2^n due to Corollary 1 (again, applied twice). Continue like this d times, ending up with a tree of depth d having 2^d leaves. The time complexity is $d2^n$.

As mentioned, with 2^d leaves, meaning 2^d new target images, finding by brute force a true preimage has complexity 2^{2n-d}. Hence, the total time complexity is about $d2^n + 2^{2n-d}$. Memory requirements are $2^{d+1} - 1$ intermediate hash values and a negligible number of message blocks.

Observe that with $d = 0$ one gets time complexity 2^{2n} and space complexity 1, which is not surprising since we do not build a tree at all, so we have a standard brute force preimage attack. With $d = n/2$ one gets time complexity about $2^{3n/2}$ and space complexity about $2^{n/2}$, equivalent to the attack of Lai and Massey, but the technique is different. The most efficient attack appears when $d = n$, in which case the time complexity is about $(n+1)2^n$, and the space complexity is 2^{n+1}. We improve the efficiency of this particular time/memory trade-off in Section 4.2.

We note that this attack provides practically any time/memory trade-off for which the product of the time and the space complexities is about 2^{2n}. Figure 5 shows some example trade-offs.

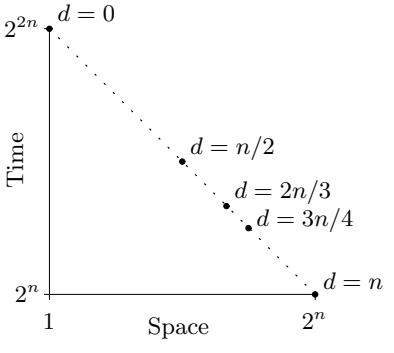


Fig. 5. A visualisation of the time/memory trade-off. Both axes are logarithmic. The case $d = 0$ corresponds to the brute force attack. Larger values of d constitute improvements with respect to attack efficiency.

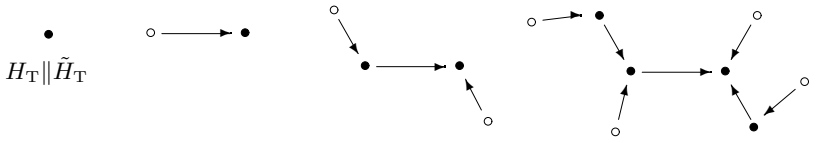


Fig. 6. Constructing a tree of pseudo-preimages by finding one child of every node in each step

Alternative methods. The tree above does, in fact, not have to be binary. If every node has on average 2^b children, then when the tree has depth d, there are 2^{bd} leaves. The time required to construct the tree is $d2^{b+n-1}$. The time required for Step 2 above is 2^{2n-bd}. The memory requirements are about 2^{bd} for reasonably large b. With $b = n/(d+1)$, which approximately balances the time spent in Steps 1 and 2, the total time complexity is about $(d/2+1)2^{n(d+2)/(d+1)}$ and the memory requirements are $2^{nd/(d+1)}$.

An alternative way of constructing the tree is the following. First, find a pseudo-preimage of the root. Then, find a pseudo-preimage of the root and its child. Continue applying Corollary 1 this way, finding in each step a pseudo-preimage for each node in the tree, thus doubling the tree size in every step. After d steps, the tree contains 2^d nodes. The time complexity is $d2^{n-1}$. See Figure 6.

Now, if there is no length padding, then we may perform a brute force search that links the initial value to any of the 2^d nodes in the tree. This brute force search has complexity 2^{2n-d}. Compared to the variant of the previous section, both time and space requirements are roughly halved. We note that this attack resembles a method described by Leurent [17] of finding preimages of MD4.

Length padding can be circumvented in the same way as it is circumvented in Kelsey and Schneier's second preimage attack on the Merkle-Damgård construction [12], but the resulting attack is slightly slower than the variant above, since there is (apparently) no efficient method of finding fixed points of the compression function.

4.2 Pushing the Time Complexity Down to 2^n

The attack above can be modified to obtain an attack of time complexity very close to 2^n. The attack applies a technique which bears some resemblance with the one used in a preimage attack by Mendel and Rijmen on the HAS-V hash function [19], and also with the P^{3}graph method introduced by De Cannière and Rechberger in [4]. The attack works as follows:

1. Choose two message blocks m_0 and m_1 arbitrarily, but with correct padding for a message of length $n+1$ blocks. Here we assume that padding does not fill an entire message block.
2. Compute $f(i, m_b)$ for each $b \in \{0, 1\}$ and for every i from 0 to $2^n - 1$. Store the outputs in the lists U_b, sorted on the output. Sorting can be done in linear time by using, e.g., Bucket-Sort or direct addressing [3].

Table 2. Time complexities of the preimage attack of Section 4.2 compared to the previous best known preimage attack of Lai and Massey, and to a brute force attack. For details on the case of DES ($n = 54$), we refer to Appendix A.2.

n	Preimage attack complexity		
	Section 4.2	Lai-Massey	Brute force
54	2^{55}	2^{81}	2^{108}
64	2^{65}	2^{96}	2^{128}
128	2^{129}	2^{192}	2^{256}
256	2^{257}	2^{384}	2^{512}

3. Construct a binary tree with 2^n leaves having the target image $H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$ as root (as above for $d = n$). The two children of each node in the tree are found by lookups in U_0 and U_1, respectively.
4. Given 2^n new target images (namely the leaves in the tree), perform a brute force search starting from the initial value of the hash function.

Step 2 above takes time 2^n. Memory requirements for each of the lists U_b are 2^n values of n bits. Step 3 is expected to take a negligible amount of time compared to Step 2, since the tree is constructed by about 2^n table lookups. Step 4 takes an expected time 2^n, since there are 2^n target images, and the probability of reaching each of them is 2^{-2n}. In total, the time complexity of the attack is about 2^{n+1}, and the memory requirements are about the same.

We note that if padding spans several message blocks, a few initial steps are required to invert through the padding blocks. This may add a small factor of 2^n to the complexity.

Table 2 shows some example complexities of this attack for different sizes of n, compared to the previous best known preimage attack and the brute force attack.

5 Multicollisions

The preimage attack described in the previous section can be used to construct multicollisions for the construction (2). Let the hash function be H, and let its initial value be $H_{\mathrm{iv}} \| \tilde{H}_{\mathrm{iv}}$. Apply the above preimage attack twice with target hash value $H_{\mathrm{iv}} \| \tilde{H}_{\mathrm{iv}}$, yielding two messages M_0 and M_1. In other words, we find M_0, M_1 such that $H(M_0) = H(M_1) = H_{\mathrm{iv}} \| \tilde{H}_{\mathrm{iv}}$. Now we can construct a 2^t-collision for arbitrary t; the messages in the multicollision consist of t copies of M_0 or M_1, concatenated together.

The time complexity is twice the complexity of the preimage attack, i.e., 2^{n+2}. For $t > 4$ this is more efficient than the previous best known multicollision attack by Joux [11], which has time complexity $t2^n$, assuming a birthday attack is used to produce each individual collision; by applying the collision attack of Section 3, the complexity is reduced to (very roughly) $(t \log_2(n)/n)2^n$. Still the

multicollision attack based on the preimage attack is faster when $t > 4n/\log_2(n)$. A drawback of the preimage-based method is memory requirements, which are about 2^{n+1} in our attack, whereas by using cycle-finding methods [6, 2], the memory requirements of Joux's attack can be reduced to a negligible quantity.

6 Other Non-random Properties

Say M is a message of t blocks, and let $H(M) = H_t \| \tilde{H}_t$ be the MDC-2 hash of M. The probability that $H_t \neq \tilde{H}_t$ is $(1 - 2^{-n})^t$, because the two halves must be different after the processing of every block out of the t blocks, in order for them to be different at the end. For an ideal $2n$-bit hash function, this probability is $1 - 2^{-n}$, irrespective of the value of t. Hence, when $t \gg 1$, the probability of the two output halves being equal is much higher in MDC-2 than in an ideal hash function. In fact, if $t = 2^n$, then the probability is around $1 - 1/e \approx 0.63$, since $(1 - 2^{-n})^{2^n} \approx 1/e$ for plausible values of n. The property does not hold for the construction (2) in general (nor does it hold if some key bits are fixed to ensure that the two keys in each iteration are different). What is required is that some n-bit value b exists for every n-bit value a such that $g(a\|a) = b\|b$.

If, during the processing of a message, one has obtained two equal halves, a standard birthday collision attack can be applied in time $2^{n/2}$. Hence, a new type of birthday attack on MDC-2 is as follows. Search for a message block m_0 such that $f(H_0, m_0) = f(\tilde{H}_0, m_0) = H_1$. Then find a pair (m_1, m_1') of message blocks such that $f(H_1, m_1) = f(H_1, m_1')$. This attack takes the same amount of time as a standard birthday attack (it is in fact faster by a factor of two, since f only has to be called 2^n times), but a naive implementation uses only $2^{n/2}$ memory compared to 2^n for a (naive) standard birthday attack. By using cycle-finding methods, memory requirements can be made negligible in both cases.

7 Application to Other Constructions

The construction (2) can be generalised even further. For example, we may define the following general construction, where f and $\tilde{f}$ are two distinct functions both mapping as $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, and $g : \{0,1\}^{2n} \to \{0,1\}^{2n}$ is (again) an invertible mapping:

$$\begin{aligned} W &= f(H_{i-1}, m_i) \| \tilde{f}(\tilde{H}_{i-1}, m_i) \\ H_i \| \tilde{H}_i &= g(W). \end{aligned} \tag{3}$$

Our attacks also apply to this construction, except that in some cases the complexity is up to twice as high. For instance, finding a pseudo-preimage of $H_\mathrm{T} \| \tilde{H}_\mathrm{T}$ now requires 2^n evaluations of both f and $\tilde{f}$, and hence the total time complexity is comparable to 2^n compression function evaluations, and not 2^{n-1} as is the case when $f = \tilde{f}$.

Apart from MDC-2 we have not found other constructions in the literature that fall under the category of (2) or (3). However, a construction that easily

comes to mind is the dual of MDC-2, meaning that the message block is used as the key in the block cipher calls, and the chaining value is used as the plaintext and also in the feed-forward. An advantage of this dual construction is that some block ciphers accept keys that are larger than the plaintext block, and hence the message blocks are larger which results in improved performance. However, since this construction is an instance of (2), it is susceptible to the attacks described in this paper.

8 Conclusion

In this paper we presented the first collision attack on the MDC-2 construction having time complexity below that of a birthday attack. The attack applies to other constructions similar to MDC-2, and does not rely on weaknesses of the underlying block cipher.

We also described new and improved time/memory trade-offs for preimage attacks, where almost any trade-off such that the product of time and space complexities is about 2^{2n}, with time complexity between 2^n and 2^{2n}, is possible. These new trade-offs mean that, e.g., a second preimage attack on MDC-2 based on DES (see Appendix A) is not far from being practical.

We showed how to construct multicollisions based on the fastest preimage attack, and we discussed some other constructions to which our attacks apply.

We believe the attacks have great theoretical and potential practical significance. Double-length schemes have been studied intensively in the last two or three decades, and for many years it was believed that MDC-2 was collision resistant, assuming the underlying block cipher was secure. In fact, the main criticism of MDC-2 seems to have been its somewhat poor performance. These attacks show that we still have a lot to learn about double-length constructions, although the recent shift towards provably secure schemes provides some consolation.

Acknowledgements

We would like to thank Prof. Bernhard Esslinger of University of Siegen, Joerg-Cornelius Schneider and Henrik Koy of Deutsche Bank for discussions on MDC-2-DES and providing us with references for its use [26,15,10]. We also would like to thank the reviewers for insightful comments and for providing the reference to Feller [5] for the estimate (1).

References

1. Brachtl, B.O., Coppersmith, D., Hyden, M.M., Matyas Jr., S.M., Meyer, C.H.W., Oseas, J., Pilpel, S., Schilling, M.: Data authentication using modification detection codes based on a public one way encryption function, March 13, 1990, US Patent no. 4,908,861. Assigned to IBM. Filed (August 28, 1987), `http://www.google.com/patents?vid=USPAT4908861` (2008/09/02)

2. Brent, R.P.: An improved Monte Carlo factorization algorithm. BIT Numerical Mathematics 20(2), 176–184 (1980)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1990)
4. De Cannière, C., Rechberger, C.: Preimages for Reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008)
5. Feller, W.: An Introduction to Probability Theory and Its Applications, 3rd edn., vol. 1. Wiley, Chichester (1968)
6. Floyd, R.W.: Nondeterministic Algorithms. Journal of the Association for Computing Machinery 14(4), 636–644 (1967)
7. Hellman, M.E.: A Cryptanalytic Time–Memory Trade-Off. IEEE Transactions on Information Theory IT-26(4), 401–406 (1980)
8. Hirose, S.: Some Plausible Constructions of Double-Block-Length Hash Functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
9. International Organization for Standardization. ISO/IEC 10118-2:1994. Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher algorithm (1994) (Revised in 2000)
10. International Organization for Standardization. ISO 9735-6:2002. Electronic data interchange for administration, commerce and transport (EDIFACT) – Application level syntax rules (Syntax version number: 4, Syntax release number: 1) – Part 6: Secure authentication and acknowledgement message (message type – AUTACK) (2002), `http://www.gefeg.com/jswg/v41/data/V41-9735-6.pdf` (2008/09/02)
11. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
12. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
13. Knudsen, L.R., Lai, X., Preneel, B.: Attacks on Fast Double Block Length Hash Functions. Journal of Cryptology 11(1), 59–72 (1998)
14. Knudsen, L.R., Preneel, B.: Fast and Secure Hashing Based on Codes. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 485–498. Springer, Heidelberg (1997)
15. Kraus, D.: Integrity mechanism in German and international payment systems (2002), `http://www.src-gmbh.de/whitepapers/Intergrity_mechanisms_in_payment_systems_Kraus_en.pdf` (2008/09/02)
16. Lai, X., Massey, J.L.: Hash Functions Based on Block Ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
17. Leurent, G.: MD4 is Not One-Way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
18. Lucks, S.: A Failure-Friendly Design Principle for Hash Functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
19. Mendel, F., Rijmen, V.: Weaknesses in the HAS-V Compression Function. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 335–345. Springer, Heidelberg (2007)
20. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
21. Meyer, C.H., Schilling, M.: Secure Program Load with Manipulation Detection Code. In: Proceedings of SECURICOM 1988, pp. 111–130 (1988)

22. Nandi, M.: Towards Optimal Double-Length Hash Functions. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 77–89. Springer, Heidelberg (2005)
23. Nandi, M., Lee, W., Sakurai, K., Lee, S.: Security Analysis of a 2/3-Rate Double Length Compression Function in the Black-Box Model. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 243–254. Springer, Heidelberg (2005)
24. National Bureau of Standards. Data Encryption Standard (DES), Federal Information Processing Standards Publication (FIPS PUB) 46 (January 15, 1977)
25. Steinberger, J.P.: The Collision Intractability of MDC-2 in the Ideal-Cipher Model. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 34–51. Springer, Heidelberg (2007)
26. Struif, B.: German Health Professional Card and Security Module Card, Specification, Pharmacist & Physician, v. 2.0 (2003), `http://www.dkgev.de/media/file/2589.spez-engl-3.pdf` (2008/09/02)
27. van Tilborg, H.C.A. (ed.): Encyclopedia of Cryptography and Security. Springer, Heidelberg (2005)
28. Viega, J.: The AHASH Mode of Operation, Manuscript (September 2004), `http://www.cryptobarn.com/papers/ahash.pdf` (2008/09/02)

A The Special Case of MDC-2 Instantiated with DES

For simplicity, throughout the paper we assumed that the key size k equals the block size n of the block cipher with which MDC-2 is instantiated. However, this is not necessarily the case, with DES [24] ($n = 64$, $k = 56$) being the most prominent example. The effective key size for MDC-2 with DES is further reduced by two bits to $k = 54$. For the following, it suffices to think of the mapping from chaining blocks to keys as a truncation from 64 to 54 bits. The exact details of this mapping are of no concern for the following treatment, hence we refer to [9] for the full details.

A.1 Collision Attacks

The collision attack as described in Section 3 produces a collision in the last chaining value of length $2n$. However, if an arbitrary message block is appended to the expected colliding message pair, it suffices to look for a collision in the $2k$ bits that will be used as the key input of DES in the following iteration. Hence, for the collision attack on MDC-2 based on DES having complexity about $2^{51.5}$, instead of two, at least three message blocks are needed.

A.2 Preimage Attacks

Also for the preimage attack of Section 4, the target hash is assumed to be of size $2n$. In order to take advantage of a smaller key size k, the last message block needs to be known by the attacker. In this case the time complexity can be as low as 2^{55}; if no first preimage is given then the attack has a complexity of about 2^{65}.

Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC*

Xiaoyun Wang[1,2], Hongbo Yu[1], Wei Wang[2],
Haina Zhang[2], and Tao Zhan[3]

[1] Center for Advanced Study, Tsinghua University, Beijing 100084, China
{xiaoyunwang,yuhongbo}@mail.tsinghua.edu.cn
[2] Key Laboratory of Cryptographic Technology and Information Security,
Ministry of Education, Shandong University, Jinan 250100, China
{wwang,hnzhang}@math.sdu.edu.cn
[3] Shandong University, Jinan 250100, China
zhantao@sdu.edu.cn

Abstract. In this paper, we present the first distinguishing attack on HMAC and NMAC based on MD5 without related keys, which distinguishes the HMAC/NMAC-MD5 from HMAC/NMAC with a random function. The attack needs 2^{97} queries, with a success probability 0.87, while the previous distinguishing attack on HMAC-MD5 reduced to 33 rounds takes $2^{126.1}$ messages with a success rate of 0.92. Furthermore, we give distinguishing and partial key recovery attacks on MDx-MAC based on MD5. The MDx-MAC was proposed by Preneel and van Oorschot in Crypto'95 which uses three subkeys derived from the initial key. We are able to recover one 128-bit subkey with 2^{97} queries.

Keywords: HMAC, NMAC, MDx-MAC, MD5, Distinguishing attack, Key recovery.

1 Introduction

Many cryptographic schemes and protocols use hash functions as primitives. In recent work [3,4,16,17,18,19], devastating collision attacks on hash functions from the MD4 family were discovered. Such attacks have undermined the confidence in the most popular hash functions such as MD5 or SHA-1, and raise the interest in reevaluating the actual security of the Message Authentication Code (MAC) algorithms based on them [7,6,9].

HMAC and NMAC are hash-based message authentication codes proposed by Bellare, Canetti and Krawczyk [1]. NMAC is the theoretical foundation of HMAC, and HMAC has been implemented in widely used protocols including SSL/TLS, SSH and IPsec. The security of NMAC and HMAC has been carefully analyzed in [1,2]. It was proved that NMAC is a pseudo-random function family (PRF) under the assumption that the compression function of the keyed hash function is a

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* Supported by the National Natural Science Foundation of China (NSFC Grant No. 60525201 and No. 90604036) and 973 Project (No.2007CB807902 and No.2007CB807903).

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 121–133, 2009.

PRF. This proof can be extended to HMAC by an additional assumption: the key derivation function in HMAC is a PRF. However, if the underlying hash function is weak (such as MD5), the above proofs may not work.

There are three types of the attacks on HMAC/NMAC, namely, distinguishing attacks, existential forgery attacks and universal forgery attacks. Distinguishing attacks can be divided into *distinguishing-R* and *distinguishing-H* attacks [9], where distinguishing-R attack means distinguishing HMAC/NMAC from a random function, and distinguishing-H attack detects instantiated HMAC/NMAC (by existing hash functions) from HMAC/NMAC with a random function. A general distinguishing-R attack on HMAC using the birthday paradox was introduced by Preneel and van Oorschot [10]. This attack requires about $2^{\frac{l}{2}}$ messages and works with probability 0.63, where l is the length of the initial value.

In this paper, we focus on the distinguishing-H attack on HMAC/NMAC-MD5 that checks which cryptographic hash function is embedded in HMAC/NMAC. For simplicity, we call it distinguishing attack. In [9], Kim *et al.* introduced two kinds of distinguishers of the HMAC structure, the differential distinguisher and the rectangle distinguisher, and used them to analyze the security of HMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. For MD5 reduced to 33 rounds, they described a distinguishing attack taking $2^{126.1}$ messages with a success probability 0.92. Using the pseudo-collisions found by den Boer and Bosselaers [5], Contini and Yin [6] proposed a related-key distinguishing attack on NMAC-MD5 with 2^{47} queries and a success probability of 0.25. They applied it to construct forgery and partial key-recovery attacks on NMAC-MD5. Fouque *et al.* [7] presented the first full-key recovery attack on HMAC/NMAC-MD4, and extended Contini and Yin's attack to the full key-recovery attack on NMAC-MD5. The latter was independently found by Rechberger and Rijmen [11,12] with better results than [7] by ignoring the conditions in the last 5 steps. They also proposed a full key-recovery attack in the related-key setting on NMAC with SHA-1 reduced to 34 steps, and improved the attack on HMAC instantiated with reduced SHA-1 variants of more steps in [12]. Recently, Wang *et al.* [15] suggested more efficient full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5 using near-collisions. However, all the attacks on NMAC-MD5 are in the related-key setting, hence these attacks can not be applied to the corresponding HMAC.

In this paper, we are able to get rid of related keys, and propose the first distinguishing attacks on HMAC/NMAC-MD5. Based on the dBB pseudo-collision [5], we search for a new kind of collision which is called a dBB collision. With the specific structure and high probability of a dBB collision, we can successfully distinguish a dBB collision from other random collisions found by the birthday attack. Once a dBB collision is detected, the distinguisher outputs HMAC/NMAC-MD5; otherwise, it outputs HMAC/NMAC with a random function. The attack needs 2^{97} queries in total, and the success rate is 0.87.

Another contribution of this paper is to introduce distinguishing and partial key recovery attacks on the MDx-MAC based on MD5 called the MD5-MAC. The MDx-MAC was proposed by Preneel and van Oorschot in Crypto'95 [10] which transforms any MD4-family hash function into a MAC algorithm. It uses

three subkeys K_0, K_1 and K_2 which are derived from an original key K. The role of K_0 and K_2 is similar to the two secret keys in the envelope MAC construction, and four 32-bit words of K_1 are added to all the step operations of four rounds respectively. The dBB collision can be used not only to distinguish MD5-MAC directly, but also to capture the full subkey K_1 which is involved in the collision path. The number of queries in the attack is about 2^{97}.

This paper is organized as follows. Background and definitions are recalled in Section 2. In Section 3, we first introduce a distinguishing attack on the keyed IV MAC, which is an adaptive chosen message attack, and then extend it to distinguish HMAC/NMAC-MD5 from HMAC/NMAC with a random function. In Section 4, we present distinguishing and key recovery attacks on MD5-MAC. Finally, Section 5 concludes the paper.

2 Background and Definitions

2.1 Notations

H	:	a hash function
$\overline{H}$	:	a hash function without padding
h	:	a compression function
$x\|\|y$	:	concatenation of the two bitstrings x and y
$+$	:	addition modular 2^{32}
$\oplus$	:	bitwise exclusive OR
$\vee$	:	bitwise OR
$\wedge$	:	bitwise AND
$\lll s$	:	left-rotation by s-bit

2.2 Brief Description of MD5

MD5 [13] is a hash function proposed by Rivest as a strengthened version of MD4. It takes an arbitrary length message and produces a 128-bit hash value. First, the input message M is padded to be $\overline{M}$, a multiple of 512 bits. Suppose the length of M in bits is l. Append the bit "1" to the end of the message followed by k "0" bits, where k is the smallest non-negative integer such that $l + 1 + k = 448 \mod 512$. Then append the 64-bit block that is equal to the number l expressed using a binary representation. The padded message $\overline{M}$ is then divided into 512-bit message blocks, i.e., $\overline{M} = (M_0, \ldots, M_{n-1})$, and processed by Merkle-Damgård iterative structure. Each iteration invokes a compression function which takes a 128-bit chaining value and 512-bit message block as inputs, and outputs a 128-bit value as the hash value of this iteration.

The compression function has four rounds. Every round has 16 steps and employs a round function. For the padded message $\overline{M}$ with n blocks, the hash function is performed n iterations in total. The k-th iteration is the following:

– Input: 512-bit message $M_{k-1}=(m_0, m_1, \cdots, m_{15})$ and a 128-bit chaining value $(A_0, B_0, C_0, D_0) = CV_{k-1}$, where CV_{k-1} is the output of $(k-1)$-th iteration.
– Step update: For $0 \le i \le 15$,

$$\begin{aligned}
A_{i+1} &= B_i + (A_i + f(B_i, C_i, D_i) + w_{4i} + c_{4i}) \lll s_{4i},\\
D_{i+1} &= A_{i+1} + (D_i + f(A_{i+1}, B_i, C_i) + w_{4i+1} + c_{4i+1}) \lll s_{4i+1},\\
C_{i+1} &= D_{i+1} + (C_i + f(D_{i+1}, A_{i+1}, B_i) + w_{4i+2} + c_{4i+2}) \lll s_{4i+2},\\
B_{i+1} &= C_{i+1} + (B_i + f(C_{i+1}, D_{i+1}, A_{i+1}) + w_{4i+3} + c_{4i+3}) \lll s_{4i+3},
\end{aligned}$$

where for $0 \le j \le 63$, w_j is one of the 16 message words, c_j and s_j are step-dependent constants, f is a round-dependent Boolean function as follows:

$$\begin{aligned}
f(x,y,x) &= (x \wedge y) \vee (\neg x \wedge z) && \text{if } 0 \le i \le 3,\\
f(x,y,x) &= (x \wedge z) \vee (y \wedge \neg z) && \text{if } 4 \le i \le 7,\\
f(x,y,x) &= x \oplus y \oplus z && \text{if } 8 \le i \le 11,\\
f(x,y,x) &= y \oplus (x \vee \neg z) && \text{if } 12 \le i \le 15.
\end{aligned}$$

– Output: $CV_k = (A_0 + A_{16}, B_0 + B_{16}, C_0 + C_{16}, D_0 + D_{16})$.

$CV_n = H(M)$ is the hash value, and CV_0 is the initial value.

2.3 Pseudo-Collisions of MD5

Our attacks are based on the dBB pseudo-collisions found by den Boer and Bosselaers [5], which satisfy the following relations:

$$h(IV, M) = h(IV', M), \tag{1}$$
$$IV \oplus IV' = (2^{31}, 2^{31}, 2^{31}, 2^{31}) = \Delta^{\mathrm{MSB}}, \tag{2}$$
$$\mathrm{MSB}(B_0) = \mathrm{MSB}(C_0) = \mathrm{MSB}(D_0), \tag{3}$$

where M is a one-block message, and MSB means the most significant bit. The probability of the dBB pseudo-collision is 2^{-46}. The specific differential path for the dBB pseudo-collision is shown in Table 1. We call the relations (2) and (3) *dBB conditions*. Now we define an important type of collision as the dBB collision:

dBB collision: A collision of two-block messages $(x\|y, x'\|y)$ is called a dBB collision if

1. Let $CV = h(IV, x)$ and $CV' = h(IV, x')$. The pair (CV, CV') satisfies the dBB conditions.
2. (CV, y) and (CV', y) compose a dBB pseudo-collision, i.e., $h(CV, y) = h(CV', y)$.

2.4 Secret Prefix MAC, HMAC and NMAC

A MAC algorithm is a hash function with a secret key K as the secondary input. HMAC and NMAC are two popular MAC algorithms which are all derived from

efficient hash functions. Another three earlier hash based MACs are constructed by the *Secret Prefix Method*, *Secret Suffix Method* and *Envelope Method.*

The secret prefix method was proposed in the 1980s, and was suggested for MD4 independently in [14] and [8]. The secret prefix MAC is constructed as:

$$\text{Secret-Prefix-MAC}_K(M) = H(K\|M).$$

If the key K (maybe padded) is a full block, secret prefix MAC is equivalent to a hash function with a secret IV, which is called the keyed IV MAC. We denote this kind of MAC construction based on MD5 as *KIMAC-MD5* which is the basic design unit for HMAC/NMAC-MD5.

The NMAC function, on input message M and a pair of 128-bit independent keys (K_1, K_2), is defined as:

$$\text{NMAC}_{(K_1,K_2)}(M) = H_{K_1}(H_{K_2}(M)).$$

In fact, the outer function acts on the output of the iterated hash function, and thus involves one iteration of the compression function. That is to say, the outer function is basically the compression function h_{K_1} acting on $H_{K_2}(M)$ which has been padded to a full block size.

Since the NMAC replaces the fixed IV in H with a secret key, this requires a modification of existing implementation of the underlying hash function. The construction of HMAC is motivated to avoid this problem, and still uses the usual fixed IV. On input message M and a single secret key K, HMAC is computed as:

$$\text{HMAC}_K(M) = H(\overline{K} \oplus opad\|H(\overline{K} \oplus ipad\|M)),$$

where $\overline{K}$ is the completion by adding "0"s of K to a full block of the iterated hash function, and *opad* and *ipad* are two one-block length constants.

Basically, HMAC_K is the same as $\text{NMAC}_{(h(\overline{K}\oplus opad),h(\overline{K}\oplus ipad))}$. For simplicity, we denote the $\text{HMAC}_K(M)$ by $H_{out}(H_{in}(M))$.

2.5 Description of MD5-MAC

The MDx-MAC was proposed by Preneel and van Oorschot in Crypto'95 [10], which converts MDx-family hash functions into MAC algorithms with a key K up to 128 bits. The underlying hash function can be any of MD5, RIPEMD, SHA, or other similar algorithms except MD4. For convenience, we denote the MDx-MAC based on MD5 as MD5-MAC.

Let $\overline{MD5}$ denote MD5 algorithm without padding. The 128-bit secret key K is expanded to three 128-bit subkeys K_0, K_1 and K_2 by the following procedure.

For $i = 0$ to 2, $K_i = \overline{MD5}(K||U_i||K)$, where U_0, U_1 and U_2 are three different 96-byte constants (See [10] for details). The MD5-MAC is then obtained from MD5 with the following modifications:

1. The initial value IV of MD5 is replaced by K_0.
2. The key K_1 is split into four 32-bit words denoted by $K_1[i]$ $(0 \le i \le 3)$ which are added to the constants used in round i of each MD5 iteration respectively.

3. Following the block containing the padding and appended length as defined by MD5, an additional 512-bit block of the following form

$$\overline{K_2} = K_2||K_2 \oplus T_0||K_2 \oplus T_1||K_2 \oplus T_2$$

is appended, where T_i $(0 \leq i \leq 2)$ are three 128-bit constants.
4. The MAC value is the leftmost m bits of the hash value.

In [10], the authors recommended $m = n/2$ for most applications. For our attack, we assume $m = n$.

3 Distinguishing Attacks on HMAC/NMAC-MD5

To describe the distinguishing attack on HMAC/NMAC-MD5, we start with a distinguishing attack on KIMAC-MD5 which is an adaptive chosen message attack.

3.1 Adaptive Chosen Message Attack on KIMAC-MD5

The core of the attack is to find a pair (x, x') whose output difference satisfies the dBB conditions, and to detect whether x and x' can lead to a dBB collision by appending 2^{47} y separately with a reasonable probability.

The adversary performs the following steps:

1. Generate a structure of 2^{66} random messages, and query the MACs of these messages. We assume that the MAC algorithm is either a KIMAC-MD5 or KIMAC with a random function (KIMAC-RF).
2. Use the birthday attack [20] to find two messages (x, x') where $(H_K(x), H_K(x'))$ satisfies the dBB conditions.
3. Let $pad(pad')$ be the padding for $x(x')$. Append 2^{47} different messages y to the messages $x\|pad$ and $x'\|pad'$ respectively, and query the MACs with the two sets of 2^{47} messages.
4. If a collision $(x\|pad\|y, x'\|pad'\|y)$ is found, output the MAC as KIMAC-MD5. Otherwise, the MAC is KIMAC-RF.

The data complexity of the attack is $2^{66} + 2 \cdot 2^{47} \approx 2^{66}$ chosen messages. Since we can use the birthday attack to search pairs satisfying dBB-conditions, the time complexity is dominated by the size of the structure in Step 1 (the data collection phase), which is about 2^{66} queries, i.e., 2^{66} MAC computations.

Now let us analyze the success probability of this attack. From the above process, we observe that, when a specific message pair (x, x') is found in step 2, our attack succeeds in the following cases. If the KIMAC is based on MD5, a message y such that $H_K(x\|pad\|y) = H_K(x'\|pad'\|y)$ is searched. Otherwise, if the KIMAC is KIMAC-RF, no collision is found. The detailed computation of the probability is as follows.

For two random messages x and x', the output difference $H_K(x) \oplus H_K(x')$ satisfies the dBB conditions with probability:

$$\frac{1}{2^{128}} \times \frac{1}{4} = \frac{1}{2^{130}}.$$

According to the birthday paradox and Taylor series expansion, no matter what kind of oracle MAC is, among the 2^{66} messages, we can find a message pair (x, x') satisfying the dBB conditions with probability

$$q \approx 1 - (1 - \frac{1}{2^{130}})^{C_{2^{66}}^2} \approx 1 - e^{-2} \approx 0.86.$$

For KIMAC-MD5, the collision in step 4 happens with higher probability 2^{-46} instead of the average probability 2^{-128}. So, when the KIMAC is based on MD5, we can find a collision among 2^{47} adaptive chosen messages in Step 4 with probability $p_1 = 1 - (1 - \frac{1}{2^{46}})^{2^{47}} \approx 0.86$. Otherwise, a collision occurs for KIMAC-RF with a low probability $p_2 = 1 - (1 - \frac{1}{2^{128}})^{2^{47}} \approx 0$. Hence, the success rate of this attack is

$$\begin{aligned} & q \times [\frac{p_1}{2} + (\frac{1 - p_2}{2})] \\ \approx\ & 0.86 \times (0.86 \times \frac{1}{2} + \frac{1}{2}) \\ \approx\ & 0.80. \end{aligned}$$

The success rate can be improved by repeating the attack several times.

3.2 Adaptive Chosen Message Attack on HMAC-MD5

The above attack cannot be applied to HMAC-MD5 directly due to the fact that the dBB collision of H_{in} is concealed by the outer level hashing H_{out}. However, we can discard all other collisions by some concrete detective techniques, and save the dBB collisions.

Suppose that we get a collision of HMAC which has the form $(x\|y, x'\|y)$. Denote $\overline{H}_{in}(x)$ as CV, and $\overline{H}_{in}(x')$ as CV', for simplicity. Let $\Delta CV = CV \oplus CV'$. The key of our attack is to distinguish the dBB collisions according to the relation of $\overline{H}_{in}(x)$ and $\overline{H}_{in}(x')$:

1. *Internal collision*: If $\Delta CV = 0$, $(x\|y, x'\|y)$ is called an internal collision.
2. *External collision*: If $\Delta CV \neq 0$, $(x\|y, x'\|y)$ is an external collision. Furthermore, when ΔCV satisfies the dBB conditions, and (CV, y) and (CV', y) compose a dBB pseudo-collision, $(x\|y, x'\|y)$ is a dBB collision. Otherwise, the collision is a non-dBB external collision.

The adversary performs as follows:

1. Generate 2^{89} one-block messages x randomly, and append a fixed 447-bit message y (taking padding into consideration) to each x. Query all the messages $x\|y$ to get their MACs.

2. Find all the collided messages $(x\|y, x'\|y)$ satisfying $\text{HMAC}_K(x\|y) = \text{HMAC}_K(x'\|y)$. Note that on average, there are 2^{49} internal collisions, 2 dBB collisions and 2^{50} non-dBB external collisions.
3. For all the collisions $(x\|y, x'\|y)$ collected in step 2 , we append $y' \neq y$ to x and x', query $(x\|y', x'\|y')$, and check if they collide. This way, the internal collisions can be detected. In the next step, we only need to distinguish the dBB collisions from the non-dBB external collisions.
4. For the remaining collisions, append 2^{47} different $y' \neq y$ to x and x', respectively, query the MACs for $x\|y'$ and $x'\|y'$, and check whether a collision occurs. Once a collision $(x\|y', x'\|y')$ is found, we conclude that the original collision $(x\|y, x'\|y)$ is a dBB collision, and output the MAC is HMAC-MD5. Otherwise, the MAC is a HMAC-RF.

Complexity evaluation

There are at most 2^{177} pairs produced by 2^{89} messages, so the expected number of internal collisions is $2^{177-128} = 2^{49}$. Similarly, the expectation of non-dBB external collisions is $2^{49} + 2^{49} = 2^{50}$ where 2^{49} collisions occur after H_{in} and other 2^{49} collisions occur after H_{out}. For two messages x and x', the output difference $h_K(x) \oplus h_K(x')$ satisfies the dBB conditions with probability 2^{-130}. Consequently, there are $2^{177-130} = 2^{47}$ pairs satisfying the dBB conditions, and about $2^{47-46} = 2$ of them are dBB collisions.

In step 1, the data complexity is 2^{89}. We keep a table of 2^{89} entries in step 2, finding $2^{49} + 2^{50} + 2$ collisions needs about 2^{89} table lookups. In step 3, the time complexity is about $2^{49} + 2^{50} + 2 \approx 2^{50.58}$ MAC computations. In step 4 , both the data and time complexity are about $(2^{50} + 2) \times 2^{47} \approx 2^{97}$.

Therefore, the total time complexity of attack is about 2^{97} MAC computations and 2^{89} table lookups, and data complexity is about 2^{97} chosen messages.

Success rate

As analyzed in section 3.1, we divide the success rate into two parts:

– If the MAC is HMAC-MD5, the attack succeeds when a dBB collision is found among $2^{50.58}$ collisions.
The probability that there exists a dBB collision among $2^{50.58}$ collisions is

$$1 - (1 - \frac{1}{2^{130+46}})^{2^{177}} \approx 1 - e^{-2} \approx 0.86.$$

The probability that the dBB collision can be detected in step 4 is about

$$1 - (1 - \frac{1}{2^{46}})^{2^{47}} \approx 1 - e^{-2} \approx 0.86.$$

Thus, if the MAC is HMAC-MD5, the attack can find a dBB collision with probability $0.86 \times 0.86 = 0.74$.
– If the MAC is HMAC-RF, the attack succeeds when no dBB collision is detected. The success probability is about

$$((1 - \frac{1}{2^{128}})^{2^{47}})^{2^{50}} \approx 1.$$

Therefore, the success rate of the whole attack is about

$$\frac{1}{2} \times 0.74 + \frac{1}{2} \times 1 = 0.87.$$

3.3 Chosen Message Attack on HMAC-MD5

In this subsection, we relax the adaptive chosen message attack to a chosen message attack. The data complexity will increase up to 2^{113}, but the table size is reduced from original 2^{89} to 2^{66} entries.

The chosen message distinguishing attack on HMAC-MD5 is described as follows:

1. Select a set of 2^{66} one-block messages x at random. Append a chosen 447-bit message y to all the x, and form a structure of 2^{66} messages $x\|y$. Choose 2^{47} different messages y to produce 2^{47} structures. Make 2^{113} MAC queries for all of the structures.
2. For each structure, fulfill the birthday attack [20] to find all the collisions $(x\|y, x'\|y)$ satisfying $\mathrm{HMAC}_K(x\|y) = \mathrm{HMAC}_K(x'\|y)$.
3. For each collision $(x\|y, x'\|y)$ found in step 2, we determine the type of the collision.
   - Check whether all the pairs $(x\|y', x'\|y')$ in other structures are collisions. If all other pairs $(x\|y', x'\|y')$ collide, then $(x\|y, x'\|y)$ is an internal collision.
   - Check whether there exists at least one y' such that $(x\|y', x'\|y')$ is a collision in another structure. If so, we conclude that $(x\|y, x'\|y)$ is a dBB collision, and the MAC is HMAC-MD5. If there is no dBB collision, the MAC is HMAC-RF.

It is clear that the attack needs about 2^{113} chosen messages. For each structure, the expectation is 8 internal collisions and 16 external collisions. So the total number of collisions in all 2^{47} structures is about $24 \times 2^{47} < 2^{52}$. For each collision, 2^{47} table lookups are needed. Therefore the time complexity is less than $2^{52} \times 2^{47} = 2^{99}$ table lookups, and the table size is 2^{66} entries.

The computation of success rate is the same as in subsection 3.2.

Application to NMAC: NMAC is a generalized version of HMAC as introduced in subsection 2.3. Since the above attack on HMAC-MD5 has no relation with the secret key, hence it can be applied to NMAC-MD5 directly.

4 Partial Key Recovery Attack on the MD5-MAC

Obviously, the distinguishing attack on HMAC/NMAC-MD5 described in Section 3 is also applicable to distinguish the MD5-MAC from the MDx-MAC based on a random functions.

It should be noted that our distinguishing attack on HMAC/NMAC-MD5 can not be extended to recover the inner keys. Even though the partial bits of

intermediate states of the second block y can be recovered using the method in [6], we can not derive the inner keys of HMAC/NMAC by the backward computation because of the existence of the first block x. For the MDx-MAC, the situation is different since its secret keys are not only involved in the beginning (IV) and the end, but also in every iteration. We are able to recover the second secret key K_1 involved in every iteration.

This process can be divided into three phases. The first phase is to find a dBB collision. Note that by the techniques described in section 3, it's easy to find a dBB collision $(x\|y, x'\|y)$ with the complexity of 2^{97} MAC computations. The second phase is to recover some bits of the intermediate states by the given dBB collision $(x\|y, x'\|y)$. The third phase is to recover the secret key K_1.

4.1 Recovering Some Bits of the Intermediate States

We can use the bit carry method of [6] to recover 255 bits of the intermediate chaining variables of the second block y. Let $y = y[0]y[1]...y[15]$ where each $y[i]$ is a 32-bit words. Table 1 lists the dBB differential path. The 6-th column of the table contains sufficient conditions that guarantee the dBB differential holds, and the last column lists the recovered 255 bits of the first round. The complexity of this part is less than $2^{46} \times 255 \approx 2^{54}$ MAC computations.

4.2 Recovering the 128-Bit Subkey K_1

We implement the *divided and conquer* attack to recover the 32-bit subkeys $K_1[0]$, $K_1[1]$, $K_1[2]$ and $K_1[3]$ separately.

1. **Recovering the 32-bit subkey** $K_1[0]$
   From Table 1, 95 bits in the first five variables A_1, D_1, C_1, B_1 and A_2 can be recovered. We guess the other 65 unknown bits, and for each guess, we compute $K_1[0]$, D_2, C_2, B_2, A_3 and D_3 successively.

$$\begin{aligned}
K_1[0] &= (A_2 - B_1) \ggg s_0 - A_1 - f_1(B_1, C_1, D_1) - y'[0] - C_0 \\
D_2 &= A_2 + (D_1 + f(A_2, B_1, C_1) + y[1] + c_1 + K_1[0]) \lll s_1 \\
C_2 &= D_2 + (C_1 + f(D_2, A_2, B_1) + y[2] + c_2 + K_1[0]) \lll s_2 \\
B_2 &= C_2 + (B_1 + f(C_2, D_2, A_2) + y[3] + c_3 + K_1[0]) \lll s_3 \\
A_3 &= B_2 + (A_2 + f(B_2, C_2, D_2) + y[4] + c_4 + K_1[0]) \lll s_4 \\
D_3 &= A_3 + (D_2 + f(A_3, B_2, C_2) + y[5] + c_5 + K_1[0]) \lll s_5
\end{aligned}$$

   If the 90 bits of D_2, C_2, B_2, A_3 and D_3 are consistent with the corresponding recovered bits in Table 1, we get the right secret key $K_1[0]$ and A_1, D_1, C_1, B_1 and A_2. In this way, we can compute all the chaining variable values A_i, B_i, C_i and D_i $(1 \le i \le 4)$ in the first round.

2. **Recovering the 32-bit subkey** $K_1[1]$
   Guess the 32-bit key $K_1[1]$. For each candidate $K_1[1]$, we compute A_i, B_i, C_i and D_i $(5 \le i \le 8)$ using the known values (A_4, D_4, C_4, B_4) and message

y. If $K_1[1]$ is the right key, then A_i, B_i, C_i, and D_i $(5 \leq i \leq 8)$ will satisfy all the 15 conditions in steps 17-33 of Table 1 with probability 1. Otherwise, A_i, B_i, C_i, and D_i $(5 \leq i \leq 8)$ will satisfy the 15 conditions with probability 2^{-15}. In this way, there are about $2^{32} \cdot 2^{-15} = 2^{17}$ candidates $K_1[1]$ left. It only needs two other dBB collisions $(x\|y', x'\|y')$ to discard the wrong $K_1[1]$, and capture the right one from the 2^{17} candidates. To find two other dBB collisions takes about 2^{47} MAC computations.

3. **Recover the 32-bit subkeys $K_1[2]$ and $K_1[3]$**
   From Table 1, we know that there is only one condition in the third round. This means that at most 33 dBB collisions (colliding pairs have the common first block (x, x')) are needed to filter the wrong keys from a 2^{32} key space and obtain the right key $K_1[2]$. Similarly, as there are 15 conditions in the 4-th round, 3 dBB collisions are required to determine the right $K_1[3]$.

Overall, the complexity of the key recovery is dominated by 2^{97} queries, which is the complexity of finding a dBB collision.

5 Conclusions

In this paper, we utilize the dBB pseudo-collisions to construct dBB collisions which have the dBB specific structure and differential path with high probability. The specific structure can be used to construct a distinguishing attack on HMAC/NMAC-MD5, with 2^{97} queries and 2^{97} MAC computations under adaptive chosen message attack. Under chosen message attacks, the complexities is up to 2^{113} queries and 2^{99} table lookups. For MD5-MAC, the specific differential path can be used to recover the subkey involved in the differential path with complexity of 2^{97} queries.

Acknowledgements. We would like to thank Christian Rechberger and three reviewers for their very helpful comments on the paper. We also hope to thank Guangwu Xu for revising the paper during his stay in Tsinghua University.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
3. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
4. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and reduced SHA-1. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)

5. den Boer, B., Bosselaers, A.: Collisions for the compression function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
6. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
7. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
8. Galvin, J.M., McCloghrie, K., Davin, J.R.: Secure management of SNMP networks. Integrated Network Management II, 703–714 (1991)
9. Kim, J.-S., Biryukov, A., Preneel, B., Hong, S.H.: On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (Extended abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
10. Preneel, B., van Oorschot, P.: MDx-MAC and building fast MACs from hash functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
11. Rechberger, C., Rijmen, V.: On authentication with HMAC and non-random properties. In: Dietrich, S., Dhamija, R. (eds.) FC 2007. LNCS, vol. 4886, pp. 119–133. Springer, Heidelberg (2007)
12. Rechberger, C., Rijmen, V.: New results on NMAC/HMAC when instantiated with popular hash functions. Journal of Universal Computer Science 14(3), 347–376 (2008)
13. Rivest, R.L.: The MD5 message digest algorithm. Request for Comments (RFC 1321), Network Working Group (1992)
14. Tsudik, G.: Message authentication with one-way hash functions. ACM Comput. Commun. Rev. 22(5), 29–38 (1992)
15. Wang, L., Ohta, K., Kunihiro, N.: New key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 237–253. Springer, Heidelberg (2008)
16. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
17. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
18. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
19. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
20. Yuval, G.: How to swindle rabin. Cryptologia 3, 187–190 (1979)

Appendix

Table 1. The dBB differential path and its corresponding sufficient conditions

step	m_i	CV	shift s_i	Output difference	Sufficient conditions	Recovered bits
-4		A_0		32		
-3		D_0		32		
-2		C_0		32	$C_{0,32} = D_{0,32}$	
-1		B_0		32	$B_{0,32} = C_{0,32}$	
1	m_0	A_1	7	32	$A_{1,32} = B_{0,32}$	32, 31, 30, ..., 8
2	m_1	D_1	12	32	$D_{1,32} = A_{1,32}$	32, 31, 30, ..., 13
3	m_2	C_1	17	32	$C_{1,32} = D_{1,32}$	32, 31, 30, ..., 18
4	m_3	B_1	22	32	$B_{1,32} = C_{1,32}$	32, 31, 30, ..., 23
5	m_4	A_2	7	32	$A_{2,32} = B_{1,32}$	32, 31, 30, ..., 8
6	m_5	D_2	12	32	$D_{2,32} = A_{2,32}$	32, 31, 30, ..., 13
7	m_6	C_2	17	32	$C_{2,32} = D_{2,32}$	32, 31, 30, ..., 18
8	m_7	B_2	22	32	$B_{2,32} = C_{2,32}$	32, 31, 30, ..., 23
9	m_8	A_3	7	32	$A_{3,32} = B_{2,32}$	32, 31, 30, ..., 8
10	m_9	D_3	12	32	$D_{3,32} = A_{3,32}$	32, 31, 30, ..., 13
11	m_{10}	C_3	17	32	$C_{3,32} = D_{3,32}$	32, 31, 30, ..., 18
12	m_{11}	B_3	22	32	$B_{3,32} = C_{3,32}$	32, 31, 30, ..., 23
13	m_{12}	A_4	7	32	$A_{4,32} = B_{3,32}$	32, 31, 30, ..., 8
14	m_{13}	D_4	12	32	$D_{4,32} = A_{4,32}$	32, 31, 30, ..., 13
15	m_{14}	C_4	17	32		
16	m_{15}	B_4	22	32	$B_{4,32} = C_{4,32}$	
17	m_1	A_5	5	32	$A_{5,32} = B_{4,32}$	
18	m_6	D_5	9	32	$D_{5,32} = A_{5,32}$	
19	m_{11}	C_5	14	32	$C_{5,32} = D_{5,32}$	
20	m_0	B_5	20	32	$B_{5,32} = C_{5,32}$	
21	m_5	A_6	5	32	$A_{6,32} = B_{5,32}$	
22	m_{10}	D_6	9	32	$D_{6,32} = A_{6,32}$	
23	m_{15}	C_6	14	32	$C_{6,32} = D_{6,32}$	
24	m_4	B_6	20	32	$B_{6,32} = C_{6,32}$	
25	m_9	A_7	5	32	$A_{7,32} = B_{6,32}$	
26	m_{14}	D_7	9	32	$D_{7,32} = A_{7,32}$	
27	m_3	C_7	14	32	$C_{7,32} = D_{7,32}$	
28	m_8	B_7	20	32	$B_{7,32} = C_{7,32}$	
29	m_{13}	A_8	5	32	$A_{8,32} = B_{7,32}$	
30	m_2	D_8	9	32	$D_{8,32} = A_{8,32}$	
31	m_7	C_8	14	32	$C_{8,32} = D_{8,32}$	
32	m_{12}	B_8	20	32		
33	m_5	A_9	4	32		
34	m_8	D_9	11	32		
35	m_{11}	C_9	16	32		
36	m_{14}	B_9	23	32		
37	m_1	A_{10}	4	32		
38	m_4	D_{10}	11	32		
39	m_7	C_{10}	16	32		
40	m_{10}	B_{10}	23	32		
41	m_{13}	A_{11}	4	32		
42	m_0	D_{11}	11	32		
43	m_3	C_{11}	16	32		
44	m_6	B_{11}	23	32		
45	m_9	A_{12}	4	32		
46	m_{12}	D_{12}	11	32		
47	m_{15}	C_{12}	16	32		
48	m_2	B_{12}	23	32	$B_{12,32} = D_{12,32}$	
49	m_0	A_{13}	6	32	$A_{13,32} = C_{12,32}$	
50	m_7	D_{13}	10	32	$D_{13,32} = B_{12,32}$	
51	m_{14}	C_{13}	15	32	$C_{13,32} = A_{13,32}$	
52	m_5	B_{13}	21	32	$B_{13,32} = D_{13,32}$	
53	m_{12}	A_{14}	6	32	$A_{14,32} = C_{13,32}$	
54	m_3	D_{14}	10	32	$D_{14,32} = B_{13,32}$	
55	m_{10}	C_{14}	15	32	$C_{14,32} = A_{14,32}$	
56	m_1	B_{14}	21	32	$B_{14,32} = D_{14,32}$	
57	m_8	A_{15}	6	32	$A_{15,32} = C_{14,32}$	
58	m_15	D_{15}	10	32	$D_{15,32} = B_{14,32}$	
59	m_6	C_{15}	15	32	$C_{15,32} = A_{15,32}$	
60	m_{13}	B_{15}	21	32	$B_{15,32} = D_{15,32}$	
61	m_4	A_{16}	6	32	$A_{16,32} = C_{15,32}$	
62	m_{11}	D_{16}	10	32	$D_{16,32} = B_{15,32}$	
63	m_2	C_{16}	15	32	$C_{16,32} = A_{16,32}$	
64	m_9	B_{16}	21	32		

Finding Preimages in Full MD5 Faster Than Exhaustive Search

Yu Sasaki and Kazumaro Aoki

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
sasaki.yu@lab.ntt.co.jp

Abstract. In this paper, we present the first cryptographic preimage attack on the full MD5 hash function. This attack, with a complexity of $2^{116.9}$, generates a pseudo-preimage of MD5 and, with a complexity of $2^{123.4}$, generates a preimage of MD5. The memory complexity of the attack is $2^{45} \times 11$ words. Our attack is based on splice-and-cut and local-collision techniques that have been applied to step-reduced MD5 and other hash functions. We first generalize and improve these techniques so that they can be more efficiently applied to many hash functions whose message expansions are a permutation of message-word order in each round. We then apply these techniques to MD5 and optimize the attack by considering the details of MD5 structure.

Keywords: MD5, splice-and-cut, local collision, hash function, one-way, preimage.

1 Introduction

Cryptographic hash functions are important primitives of cryptographic techniques, which generate short-length strings from arbitrary length input messages. There are many applications to make a scheme secure using a hash function: message compression in digital signatures and message authentication, for example. A hash function H should satisfy several security properties such as

- **Preimage resistance:** for given y, x *s.t.* $H(x) = y$ must be difficult to find,
- **2nd-preimage resistance:** for given x, x' *s.t.* $H(x) = H(x'), x \neq x'$ must be difficult to find,
- **Collision resistance:** A pair of (x, x') *s.t.* $H(x) = H(x'), x \neq x'$ must be difficult to find.

For a given n-bit hash value y, if the hash values of 2^n distinct messages are computed, there is a high probability that one of those values will match with y. Therefore, any method that can find a preimage faster than 2^n hash computation is a threat for hash functions. We stress that National Institute of Standards and Technology (NIST) requires preimage resistance with a complexity of 2^n for SHA-3 candidates [15].

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 134–152, 2009.

MD5 [11] was proposed by Rivest in 1991. It generates 128-bit hash values by iteratively applying a compression function consisting of 64 steps. Though its security is suspect, MD5 is one of the most widely used hash functions in the world. So, a detailed analysis of the preimage resistance of MD5 is required.

Variants of collision attacks on MD5 were proposed by den Boer and Bosselaers in 1993 [5] and by Dobbertin in 1996 [6]. The first collision attack on MD5 was proposed by Wang et al. in 2005 [16]. Since then, several improved collision attacks have been proposed. The most effective attack, proposed by Klima [8], can generate a collision in one minute with a standard PC. Although there have been several powerful collision attacks on MD5, the preimage resistance of MD5 has not been broken yet.

1.1 History of Preimage Attacks on MD4-Family

The history of preimage attacks on MD4-based hash functions is as follows. (In this paper, we omit the unit of complexity, which is the computational complexity of the compression function of the corresponding hash function.)

The first successful preimage attack was the one proposed by Leurent on MD4 at FSE 2008. The attack, with a complexity of $2^{100.5}$, generates a preimage [9]. The first preimage attack on MD5 was presented by De et al. in 2007. It attacked the first 26 steps with a SAT solver [4]. At ACISP 2008, Sasaki and Aoki presented a preimage attack with a complexity of 2^{96} on intermediate 44 steps of MD5 [13]. The paper shows that if the round order of MD5 is modified, intermediate 51 steps can be attacked. At SAC 2008, Aumasson et al. proposed a preimage attack with a complexity of 2^{102} on the first 47 steps of MD5 and an attack on full HAVAL-3 [2]. Also at SAC 2008, Aoki and Sasaki [1] showed an attack with a complexity of 2^{121} on the last 63 steps of MD5, and showed how to find a preimage of full MD5 slightly faster than the preimage resistance complexity 2^{128} by using a clever brute force algorithm. They also show one-block preimage attack on MD4. At CRYPTO 2008, Cannière and Rechberger attacked 49 steps of SHA-0 and 44 steps of SHA-1 [3]. Sasaki and Aoki proposed attacks on intermediate 52 steps of HAS-160 at ICISC 2008 [12] and 3-, 4-, and 5-pass HAVAL at Asiacrypt 2008 [14].

So far, the preimage resistance of full MD4, full HAVAL-3 and full HAVAL-4 were broken. Although these attacks are theoretically very interesting, they are not important from the industrial view point. On the other hand, regarding widely used hash functions such as MD5 or SHA-1, only step-reduced versions are analyzed and no preimage attack on the full specification has been conducted.

1.2 Related Techniques

Here, we explain previous attack techniques related to our work. See Section 3 for details of each technique.

The attacks on full HAVAL-3 and 47-steps MD5 by Aumasson et al. [2], which are later generalized by Sasaki and Aoki [14] use the *local-collision technique*,

where the *absorption properties* are used to make a local collision and the *consistency check* is performed in the attack. On the other hand, the attacks on one-block MD4 and 63-steps MD5 by Aoki and Sasaki [1] use the *splice-and-cut technique*, where the attacks are made to be more efficient with the *partial-matching technique* and the *partial-fixing technique*.

1.3 Our Results

This paper proposes the first cryptanalytic preimage attack on the full MD5. It finds a pseudo-preimage of full MD5 with a complexity of $2^{116.9}$ and a preimage of full MD5 with a complexity of $2^{123.4}$. The memory complexity of the attack is $2^{45} \times 11$ words.

In this paper, first, we improve several existing techniques with respect to following four points so that they can be more efficiently applied to various hash functions.

1. **Generalization of the local-collision technique**
   As described in a previous paper [14], the local-collision technique can be applied only if the two chosen neutral words are located a certain number of steps away. Since this limitation is too restrictive, the local-collision technique could not be applied to full MD5. Another paper [12] shows a variant of the local-collision technique; however, it is particular to HAS-160, which is the attack target of the paper. In this paper, we generalize the local-collision technique so that the same advantage can be obtained in various situations. Because our new technique no longer forms a local-collision, we call it *initial-structure technique*.
2. **Extension of the absorption properties**
   When we construct the initial structure, the absorption properties must be considered. In this paper, we newly consider *cross absorption properties*, which are extended versions of the absorption properties. This can further increase the situations where the initial structure can be constructed.
3. **Partial-fixing technique for unknown carry behavior**
   The partial-fixing technique partially computes the step function even if a part of the message words and chaining variables are not known. In previous papers, only partial computations whose carried number effects are deterministic were considered. In this paper, we also consider partial computations where an attacker cannot guess the carried number behavior in advance. Then, we propose an efficient attack procedure that does not increase the total attack complexity.
4. **Efficient consistency check method**
   We also solve a problem of an existing technique, where the consistency of the initial structure or the local collision is inefficiently checked, and thus the complexity becomes too high to attack successfully in some situation.

We stress that our improved techniques are not particular to MD5, but can be generally applied to hash functions whose message expansions are permutations of message-word order in each round.

Table 1. Comparison of preimage attacks on MD5

Paper	Number of attacked steps	Complexity	
		Pseudo-preimage	Preimage
[4]	26	Not given †	
[13]	44 (Steps 3-46)	2^{96} †	
[2]	47	2^{96}	2^{102}
[1]	63	2^{112}	2^{121}
[1]	64 (Full)	$2^{125.7}$ ‡	2^{127} ‡
This paper	64 (Full)	$2^{116.9}$	$2^{123.4}$

† One-block attack.
‡ The brute force attack is used, but the computation order is optimized.

Secondly, we combine all of our improved techniques and apply them to full MD5. Then, we optimize the attack by considering the details of MD5 structure. A summary of our results and previous results is shown in Table 1.

The organization of this paper is as follows. In Section 2, we describe the specification of MD5 and introduce the notation. In Section 3, we briefly describe the related work. In Section 4, we improve several existing techniques. In Section 5, we describe the attack on MD5 in detail and evaluate its complexity. In Section 6, we conclude this paper.

2 Description of MD5

2.1 MD5 Specification and Its Properties

This section describes the specification of MD5. For details, we refer to [11].

MD5 is one of the Merkle-Damgård hash functions, that is, the hash value is computed as follows:

$$\begin{cases} H_0 \leftarrow IV, \\ H_{i+1} \leftarrow \mathrm{md5}(H_i, M_i) \qquad \text{for } i = 0, 1, \ldots, n-1, \end{cases} \tag{1}$$

where IV is the initial value defined in the specification, md5: $\{0,1\}^{128} \times \{0,1\}^{512} \rightarrow \{0,1\}^{128}$ is the compression function of MD5, and H_n is the output of the hash function. Before (1) is applied, the messages string M is processed as follows.

- The messages are padded in 512-bit multiples.
- The padded string includes the length of the message represented by 64 bits. The length string is represented as little endian and is placed at the end of the padding part.

After this process, the message string is divided into 512-bit blocks, M_i ($i = 0, 1, \ldots, n-1$).

Table 2. Boolean functions, rotation numbers, and message expansion of MD5

$\Phi_0, \Phi_1, \ldots, \Phi_{15}$	$\Phi_j(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$														
$\Phi_{16}, \Phi_{17}, \ldots, \Phi_{31}$	$\Phi_j(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$														
$\Phi_{32}, \Phi_{33}, \ldots, \Phi_{47}$	$\Phi_j(X,Y,Z) = X \oplus Y \oplus Z$														
$\Phi_{48}, \Phi_{49}, \ldots, \Phi_{63}$	$\Phi_j(X,Y,Z) = Y \oplus (X \vee \neg Z)$														
$s_0, s_1, \ldots, s_{15}$	7	12	17	22	7	12	17	22	7	12	17	22	7	12	17	22
$s_{16}, s_{17}, \ldots, s_{31}$	5	9	14	20	5	9	14	20	5	9	14	20	5	9	14	20
$s_{32}, s_{33}, \ldots, s_{47}$	4	11	16	23	4	11	16	23	4	11	16	23	4	11	16	23
$s_{48}, s_{49}, \ldots, s_{63}$	6	10	15	21	6	10	15	21	6	10	15	21	6	10	15	21
$\pi(0), \pi(1), \ldots, \pi(15)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(16), \pi(17), \ldots, \pi(31)$	1	6	11	0	5	10	15	4	9	14	3	8	13	2	7	12
$\pi(32), \pi(33), \ldots, \pi(47)$	5	8	11	14	1	4	7	10	13	0	3	6	9	12	15	2
$\pi(48), \pi(49), \ldots, \pi(63)$	0	7	14	5	12	3	10	1	8	15	6	13	4	11	2	9

The compression function $H_{i+1} \leftarrow \mathrm{md5}(H_i, M_i)$ is computed as follows.

1. M_i is divided into 32-bit message words m_j $(j = 0, 1, \ldots, 15)$.
2. The following recurrence is done.

$$\begin{cases} p_0 \leftarrow H_i \\ p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) \qquad \text{for } j = 0, 1, \ldots, 63 \end{cases}$$

3. H_{i+1} $(= p_{64} + H_i)$ is output, where "+" denotes 32-bit word-wise addition. In this paper, we similarly use "−" to denote 32-bit word-wise subtraction.

R_j is the step function for Step j. Let Q_j be a 32-bit value that satisfies $p_j = (Q_{j-3}, Q_j, Q_{j-1}, Q_{j-2})$. $R_j(p_j, m_{\pi(j)})$ computes p_{j+1} as follows:

$$\begin{cases} Q_{j+1} \leftarrow Q_j + (Q_{j-3} + \Phi_j(Q_j, Q_{j-1}, Q_{j-2}) + m_{\pi(j)} + k_j) \lll s_j, \\ p_{j+1} \leftarrow (Q_{j-2}, Q_{j+1}, Q_j, Q_{j-1}), \end{cases}$$

where Φ_j, k_j, and $\lll s_j$ are the bitwise Boolean function, constant value, and left rotation defined in the specification, respectively. $\pi(j)$ is a function for MD5 message expansion. Details of Φ_j, s_j, and $\pi(j)$ are shown in Table 2. Note $R_j^{-1}(p_{j+1}, m_{\pi(j)})$ is computed with almost the same complexity as that of R_j.

3 Related Works

3.1 Converting Pseudo-Preimages to a Preimage

For a given hash value H_N and a compression function CF, pseudo-preimage is a pair of $(v, M), v \neq IV$ such that $CF(v, M) = H_N$. First, we describe the generic algorithm for the Merkle-Damgård hash functions with n-bit output, which converts pseudo-preimages to a preimage [10, Fact 9.99]. Assume that there is an algorithm that finds $(H_1, (M_1, M_2, \ldots, M_{N-1}))$ such that $H_{i+1} = CF(H_i, M_i)$ $(i = 1, 2, \ldots, N-1)$ with the complexity of 2^x and H_1 looks random. Prepare

a table that includes $2^{n/2-x/2}$ entries of $(H_1, (M_1, M_2, \ldots, M_{N-1}))$. Compute $2^{n/2+x/2}$ $CF(H_0, M_0)$ for random M_0. One of the results will agree with one of the entries in the table with high probability. The required complexity of the attack is about $2^{n/2+1+x/2}$. Therefore, showing how to find (H_1, M_1) from a given hash value within $2^x, x < n-2$ is enough for a theoretical preimage attack.

3.2 Preimage Attack on 63 Steps of MD5

At SAC 2008, Aoki and Sasaki proposed a preimage attack on 63 steps of MD5 based on the splice-and-cut, partial-matching, and partial-fixing techniques [1].

The *splice-and-cut technique* is a way to apply the meet-in-the-middle attack. The authors consider the first and last steps as consecutive steps, and divide the attack target into two *chunks* of steps so that each chunk includes independent message words from the other chunk. Such message words are called *neutral words*. Then, a pseudo-preimage is computed by the meet-in-the-middle attack.

The *partial-matching technique* enables an attacker to skip several steps of an attack target when searching for chunks. Assume that one of the divided chunks provides the value of p_i, where $p_i = (Q_{i-3}, Q_i, Q_{i-2}, Q_{i-1})$, and the other chunk provides the value of p_{i+3}, where $p_{i+3} = (Q_i, Q_{i+3}, Q_{i+2}, Q_{i+1})$. p_i and p_{i+3} cannot be directly compared; however, a part of the values, that is, 32-bits of Q_i, can be compared immediately. In such a case, one can ignore messages used in steps $i, i+1$, and $i+2$ when the meet-in-the-middle attack is performed.

The *partial-fixing technique* enables an attacker to skip more steps. The idea is to fix a part of the neutral words so that an attacker can partially compute a chunk even if a neutral word for the other chunk appears. This enables the attacker to skip more steps. For example, consider the equation for computing Q_{j-3} in the inversion of the step function $R_j^{-1}(p_{j+1}, m_{\pi(j)})$:

$$Q_{j-3} = ((Q_{j+1} - Q_j) \ggg s_j) - \Phi_j(Q_j, Q_{j-1}, Q_{j-2}) - m_{\pi(j)} - k_j. \quad (2)$$

When the lower n bits of Q_{j-1}, Q_{j-2}, and $m_{\pi(j)}$ are fixed and other variables are fully fixed, the lower n bits of Q_{j-3} can be computed independently from the higher $32-n$ bits of Q_{j-1}, Q_{j-2}, and $m_{\pi(j)}$.

3.3 Preimage Attack on HAVAL

A combination of the meet-in-the-middle and local collision was first proposed by Aumasson et al. [2]. Sasaki and Aoki further improved this by using the splice-and-cut technique instead of the simple meet-in-the-middle attack [14]. As a result, they succeeded in attacking full HAVAL-3, full HAVAL-4, and step-reduced HAVAL-5, and slightly improved the complexity of the brute force attack on full HAVAL-5.

The *local-collision technique* named by Sasaki and Aoki [14] enables an attacker to skip several steps at the beginning of chunks. The key idea of this technique is to select two neutral words that can form a local collision. Schematic explanation is shown in the left diagram of Figure 1. To achieve this, the selected neutral words must be exactly $(L \cdot n + 1)$ steps away each other, where

$n \geq 1$ and L denotes the number of chaining variables, e.g., L=8 for HAVAL and L=4 for MD5. Changes of the neutral words' values must be guaranteed not to give any influence to other chaining variables. To stop the difference propagating through Φ_j, the values of the other chaining variables must be fixed so that input differences are ignored in the output value. Such properties are called *absorption properties*. Finally, changes of neutral-words' values are checked to be offset each other. For example, in the left diagram of Figure 1, we need to check $Q_{j-3}^{1\text{st}} + m^{2\text{nd}} + m^{1\text{st}} \stackrel{?}{=} Q_{j+5}^{2\text{nd}}$ for a given $(m^{1\text{st}}, Q_{j-3}^{1\text{st}}, m^{2\text{nd}}, Q_{j+5}^{2\text{nd}})$. We call such a checking procedure *consistency check*.

Because a local collision of HAVAL can be formed by only two message words and Φ_j has many absorption properties, the local-collision technique can be effectively applied to HAVAL.

3.4 Preimage Attack on HAS-160

An example of a variant of the local-collision technique is shown in Ref. [12]. Differently from Ref. [14], Ref. [12] applies the local-collision technique even if two neutral words are located three steps away, not $(L \cdot n + 1)$ steps away. However, this technique is particular to their attack target HAS-160.

4 Improved Techniques

We applied all the previously mentioned techniques to full MD5, but the attempt failed. To attack MD5, further improvements are necessary. In this section, we give some intuition of our improved idea. For the concrete application to MD5, we refer to Section 5.

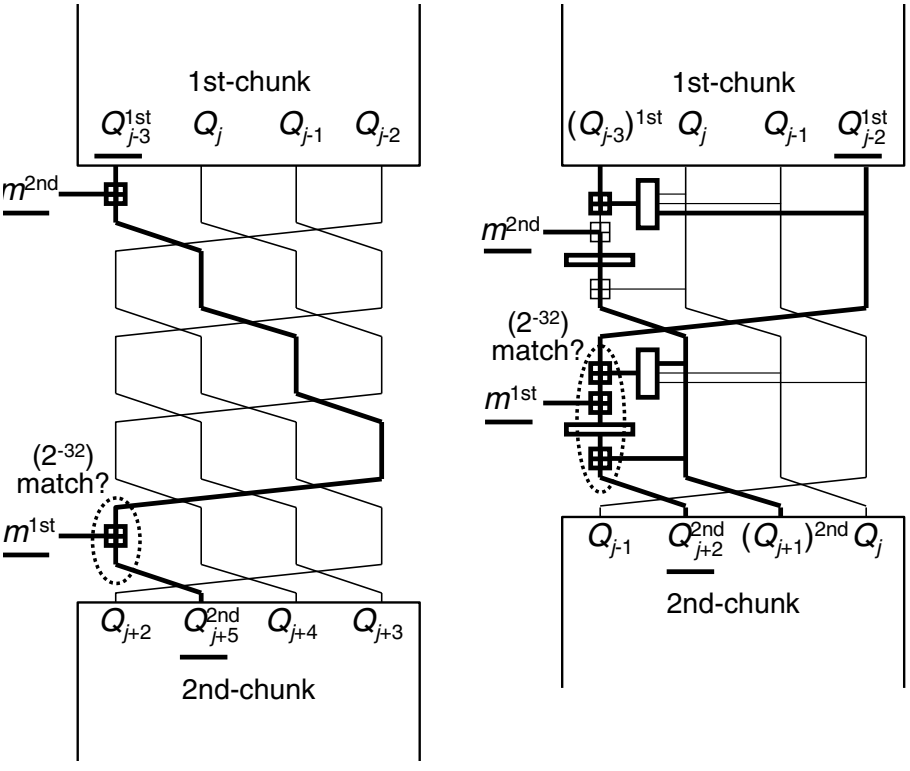
4.1 Initial Structure: Generalization of Local-Collision Technique

In the previous works, the local-collision technique is applicable if selected neutral words are exactly $(L \cdot n + 1)$ steps away. However, this technique has the following three problems.

1. The limitation of $(L \cdot n + 1)$ steps away is too restrictive to find good chunks.
2. If more than two message words are necessary to form a local collision, the limitation becomes much stronger. In fact, MD5 needs three words.
3. Absorption properties of Φ are necessary to obtain a local collision, however, Φ does not always have such properties.

The above problems are solved by our new technique. It is visualized in Figure 1.

Previous work fixes the value of Q_j, Q_{j-1}, Q_{j-2} and $Q_{j+4}, Q_{j+3}, Q_{j+2}$ for any value of $m^{1\text{st}}, Q^{1\text{st}}, m^{2\text{nd}}, Q^{2\text{nd}}$. However, we found the essential point is to make the first chunk independent of $(m^{2\text{nd}}, Q^{2\text{nd}})$ and make the second chunk independent of $(m^{1\text{st}}, Q^{1\text{st}})$. Therefore, Q_j, Q_{j-1}, Q_{j-2}, which are included in the first chunk, can be changed depending on the value of $(m^{1\text{st}}, Q^{1\text{st}})$. Similarly, $Q_{j+4}, Q_{j+3}, Q_{j+2}$ can be changed depending on $(m^{2\text{nd}}, Q^{2\text{nd}})$.



Left side describes local-collision technique; right side describes our generalization called initial structure. Underlined variables are neutral words. Notation $(Q)^x$ denotes a chaining variable whose value changes depending on the value of neutral words for x-chunk.

Fig. 1. MD5 structures with old and new techniques applied

Based on this observation, we can construct several new patterns of "local collision". Because these patterns no longer form a local collision, we call this technique *initial structure*. The following is the concept of the initial structure.

> *Initial structure is a few consecutive steps including at least two neutral words named m^{2nd} and m^{1st}, where steps after the initial structure (2nd chunk) can be computed independently of m^{1st} and steps before the initial structure (1st chunk) can be computed independently of m^{2nd}.*

In the above concept, if m^{1st} appears in an earlier step than m^{2nd}, the structure can be included in the first and second chunks, hence the attack can be easily performed. We are interested in the case where m^{2nd} appears earlier than m^{1st}.

An example of the initial structure of MD5 consisting of two steps is shown in Figure 1. In this structure, 2^{32} values of $m^{1st}, Q^{1st}, m^{2nd}, Q^{2nd}$ are tried when we compute two chunks. To make the second chunk independent of Q^{1st}, we choose

Table 3. Possible patterns of initial structure of MD5

	$m_{\pi(i)}$	$m_{\pi(i+1)}$	$m_{\pi(i+2)}$	$m_{\pi(i+3)}$	$m_{\pi(i+4)}$
Pattern 1	∘	∘			
Pattern 2	∘	∘	∘		
Pattern 3	∘	∘			∘
Pattern 4	∘			∘	∘

∘ denotes a neutral word that is necessary to form initial structure.
Note: Pattern 3 is the local collision usually considered.

Q_{j-3} to cancel the change of $Q^{1\text{st}}$. Similarly, when we compute the second chunk, we compute Q_{j+1} according to the value of $m^{2\text{nd}}$. In the end, this structure provides 2^{64} free bits for both the first and second chunks, guarantees that the first and second chunks are independent of each other, and succeeds with a probability of 2^{-32} for randomly given $m^{1\text{st}}, Q^{1\text{st}}, m^{2\text{nd}}, Q^{2\text{nd}}$.

As the example shown in Figure 1, some initial structure patterns do not use the absorption properties of Φ. This gives a big advantage to an attacker compared to the previous local-collision technique because such structures can be constructed even if Φ does not have absorption properties, e.g., Φ is XOR.

We manually searched for patterns of initial structures that are formed within 5 steps, and found that patterns shown in Table 3 can form the initial structure.

4.2 Cross Absorption Property

The cross absorption property is an extension of the absorption property. By considering the cross absorption property, the number of possible initial structure patterns can be increased.

The absorption properties of MD5 summarized in Ref. [13] focus on how to ignore one of the input variables of $\Phi_j(X, Y, Z)$. This enables us to fix the output of $\Phi_j(X, Y, Z)$ even if one of X, Y, Z changes.

Cross absorption properties enable us to fix the output of $\Phi_j(X, Y, Z)$ even if two of X, Y, Z are changed. To achieve cross absorption properties, we partially fix changing variables so that fixed bits cover all 32 bits. For example, let us consider $\Phi_j(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$, where Y and Z are neutral words. To fix the output, we first fix lower n bits of Y and fix lower n bits of X to 1. Then, we fix higher $32 - n$ bits of Z and fix higher $32 - n$ bits of X to 0. As a result, the output of Φ_j is fixed to Y in lower n bits and Z in higher $32 - n$ bits for any value of lower n bits of Z and higher $32 - n$ bits of Y.

By considering the cross absorption properties, more complicated initial structures shown in Table 4 can be constructed. Pattern 6 is useful because only two

Table 4. Initial structure with cross absorption properties

	$m_{\pi(i)}$	$m_{\pi(i+1)}$	$m_{\pi(i+2)}$	$m_{\pi(i+3)}$	$m_{\pi(i+4)}$
Pattern 5	∘		∘		
Pattern 6	∘			∘	

message words are involved and the length of the structure is relatively long (4 steps). See Section 5.2 for the application to MD5.

4.3 Partial-Fixing Technique for Unknown Carried Number Behavior

The previous partial-fixing technique on MD5 [1] enables us to skip six steps at the end of chunks by partially computing the chaining variables. Let us consider the equation $A + B$, where A and B are only partially known. When known part of A and B starts from LSB, we can uniquely compute $A + B$ in the same number of bits, whereas, when known part starts from an intermediate bit x, we cannot uniquely determine intermediate bits of $A + B$ due to the unknown carried number from bit $x-1$ to x. However, by considering both possible carried number patterns, the number of candidates of $A + B$ can be reduced to only two. Consequently, for each addition of values with intermediate known bits, we obtain the correct pairs and the same number of wrong pairs.

A small amount of incorrect pairs can be filtered out with negligible complexity. After we find the corresponding message by a partial matching test, we compute the step function and check the exact carried number value step by step. This computation costs only 1 step, that is, $2^{-6}(= \frac{1}{64})$ MD5 computations, and the number of remaining pairs will be reduced by checking the correctness of carried number assumption and matching test for increased known bits. In the end, when the number of unknown carried numbers is up to 6, we consider all 2^6 possible carried number patterns, and incorrect data is filtered out with a complexity of $1(= 2^6 \cdot 2^{-6})$ MD5 computations, which is a very small extra cost. This enables us to skip eight steps at the end of chunks. See Section 5.3 for the application to MD5.

4.4 Efficient Consistency Check Method

In previous works, the consistency of the initial structure (or local-collision) is checked after the partial matching test of chunk's results is finished. This strategy fails if the number of matched bits is small.

For example, we consider the attack procedure for the left structure in Figure 1. We compute chunks for 2^{64} values of $(m^{1\text{st}}, Q^{1\text{st}})$ and $(m^{2\text{nd}}, Q^{2\text{nd}})$. Assume the partial matching test works for small numbers of bits, e.g., only 12 bits. Ideally, we should obtain $2^{84}(= 2^{128} \cdot 2^{-32} \cdot 2^{-12})$ pairs where the partial 12 bits are matched and the initial structure is properly satisfied with a complexity of 2^{84}. Therefore, by repeating the above procedure 2^{32} times, we expect to obtain a pair where all 128 bits are matched with a complexity of $2^{116}(= 2^{84} \cdot 2^{32})$. However, the previous method computes a few steps for $2^{116}(= 2^{128} \cdot 2^{-12})$ pairs after the 12-bit matching test, and then, checks the full-bit match test and finally checks the consistency of the initial structure, which is satisfied with a probability of 2^{-32}. Computing a few steps for 2^{116} pairs costs roughly 2^{116} step function computation, and repeating this procedure 2^{32} times requires 2^{148}, which is worse than the brute force attack.

We solve this problem by performing the consistency check together with the partial matching test. This can be performed with a small amount of extra computation and memory. Again, we consider the attack procedure for the left structure in Figure 1. When we compute the first chunk by trying all (m^{1st}, Q^{1st}), we additionally store the value of $m^{1st} + Q^{1st}$ in a table. Then, when we compute the second chunk by trying all (m^{2nd}, Q^{2nd}), we compute $Q^{2nd} - m^{2nd}$ and compare it with $m^{1st} + Q^{1st}$ stored in the table. By this effort, the previous example examines a 44-bit matching test instead of a 12-bit matching test for 2^{128} pairs, and thus, the complexity becomes 2^{84}. After 2^{32} repetation of this procedure, we will find a 128-bit matched pair with a complexity of 2^{116}.

5 Preimage Attacks on Full MD5

5.1 Selected Initial Structure and Chunks

When we searched for good chunks, we assumed that the partial-matching and partial-fixing techniques could enable us to skip a maximum of 11 steps. Under this assumption, we considered all possible patterns and positions of the initial structure. As a result, we found that the pattern 6 in Table 4 for $i = 14$ skipping steps 43-50 is the only useful pattern. This chunk separation is shown in Figure 2.

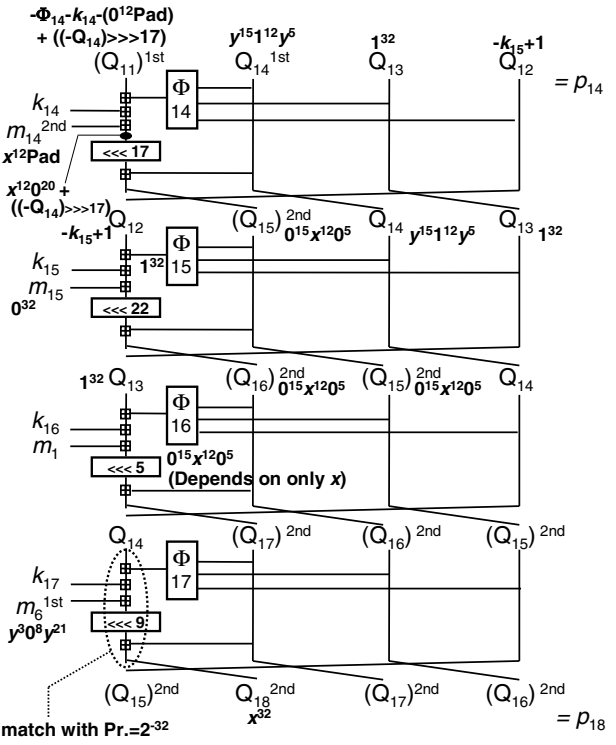
5.2 Details of Initial Structure for Full MD5

Construction of the initial structure is complicated. We need to consider the rotation number s_j and constant k_j in each step. First, we show how to fix message words and chaining variables inside the initial structure in Figure 3 and then explain how computations in the initial structure behave. We have confirmed that the numbers and positions of fixed bits are optimal when both of the initial structure and partial-fixing techniques are considered.

Numbers written in a small bold font near variables denote the value of each variable. To denote bit information of variables, we use notation a^b, which means

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index	0	1	2	3	4	5	⑥	7	8	9	10	11	12	13	⑭	15
	first chunk														initial	
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index	1	⑥	11	0	5	10	15	4	9	⑭	3	8	13	2	7	12
	structure		second chunk													
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index	5	8	11	⑭	1	4	7	10	13	0	3	⑥	9	12	15	2
	second chunk											skip				
Step	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
index	0	7	⑭	5	12	3	10	1	8	15	⑥	13	4	11	2	9
	skip		first chunk													

Fig. 2. Selected chunks for full-round MD5



The lower 20 bits of m_{14} are fixed to satisfy the message padding. For example, all bits are fixed to 1.

Fig. 3. Initial structure for full MD5

the one-bit value a continues for b bits. For example, 0^{32} means all 32 bits are 0, and $0^{15}x^{12}0^5$ means that the first 5 bits[1] are fixed to 0, the next 12 bits are free-bits for the second chunk, and the last 15 bits are fixed to 0.

To construct the initial structure, we firstly choose m_6 and Q_{14} as neutral words for the first chunk and m_{14} and Q_{18} as neutral words for the second chunk so that both chunks can produce 2^{64} items whereas the consistency of the initial structure checked in the dotted circle is satisfied with a probability of 2^{-32}. In Figure 3, we use notation ^{1st} and ^{2nd} to denote neutral words for the first and second chunks, respectively. Let x and y represent a free bit in the neutral words for the second and first chunks, respectively. Here, free bit means the unfixed bits of neutral words where we try all values when we perform the meet-in-the-middle attack. We secondly fix values of variables to guarantee that p_{14} can be computed independently of the value of 'x's and p_{18} can be computed independently of the value of 'y's. We also choose variables that are computed

[1] In this paper, LSB is the first bit (= 0th bit), and MSB is the last bit (= 31st bit).

depending on the value of neutral words for each chunk. In Figure 3, we indicate such variables with notation $(\ )^{1\text{st}}$ and $(\ )^{2\text{nd}}$.

In Remarks of this section, we will explain Q_{11} can be computed independently of 'x's of m_{14}. Therefore, p_{14} is independent of 'x's. Now, we explain why p_{18} is guaranteed to be independent of 'y's by fixing values as shown in Figure 3.

Values of 'y's in m_6 only give impact to the data line where the consistency is checked in step 17 with a probability of 2^{-32}. Therefore, p_{18} is independent of 'y's in m_6. The remaining work is to guarantee that values of 'y's in Q_{14} do not impact other data lines in steps 14, 15, and 16.

1. In step 14, values of y in Q_{14} can impact the value of Q_{15} through Φ_{14} and through the direct addition from Q_{14} to Q_{15}. To prevent these impacts, we choose the value of Q_{11} so that the sum of Q_{11}, output of Φ_{14}, $Q_{14} \ggg s_{14}$, and fixed part (lower 20 bits) of m_{14} are always the same value. Therefore, every time we choose Q_{14}, we compute Q_{11} as follows.

$$Q_{11} = -\Phi_{14}(Q_{14}, Q_{13}, Q_{12}) - k_{14} - (m_{14} \wedge \texttt{0xfffff}) + ((-Q_{14}) \ggg s_{14}), \quad (3)$$

   where we also cancel the addition of k_{14} for simplicity. This cancellation may fail because of the relationship of addition and rotation. This problem is solved in the Remarks of this section.
2. In step 15, we arrange the values of Q_{15}, Q_{14}, and Q_{13} so that changes of 'y's in Q_{14} is absorbed in the computation of Φ_{15}. Because two input variables Q_{15} and Q_{14} have free-bits, we use the cross absorption property introduced in Section 4.2. Remember $\Phi_{15} = (Q_{15} \wedge Q_{14}) \vee (\neg Q_{15} \wedge Q_{13})$. Because the values of Q_{15} and Q_{13} are 0 and 1, respectively, in bit positions 0-4 and 17-31, the value of Φ_{15} becomes 1. In bit positions 5-16, because the values of Q_{14} and Q_{13} are 1, the value of Φ_{15} becomes 1. Therefore, regardless of the value of 'y's in Q_{14}, the output of Φ_{15} is fixed to 1^{32}.
3. In step 16, the Boolean function is $\Phi_{16} = (Q_{16} \wedge Q_{14}) \vee (Q_{15} \wedge \neg Q_{14})$. If Q_{16} can be fixed to the same value as Q_{15}, Q_{14} is absorbed in the computation of Φ_{16}. This is achieved by setting $Q_{12} + \Phi_{15} + k_{15} + m_{15} = 0$ since $Q_{16} = Q_{15} + (Q_{12} + \Phi_{15} + k_{15} + m_{15}) \lll 22$. Remember, m_{15} is involved in the message padding part. To guarantee that the length of the preimage will be at most $2^{32} - 1$ bits, we fix m_{15} to 0. We know that $\Phi_{15} = \texttt{0xffffffff} = -1$. Therefore, fixing $Q_{12} = -k_{15} + 1$ can achieve the desired condition.

Finally, p_{18} is guaranteed to be independent of 'y's, and the initial structure is properly constructed for any selection of 'x's and 'y's.

Remarks. Computation for step 14 performed by equation (3) may fail and the probability of this depends on the values of chaining variables and the message word. We experimentally confirmed that for all 2^{32} possible patterns of unfixed bits in (m_{14}, Q_{14}), the choice of Q_{14} does not impact m_{14} with high probability. Specifically, for any (m_{14}, Q_{14}), the following equation holds.

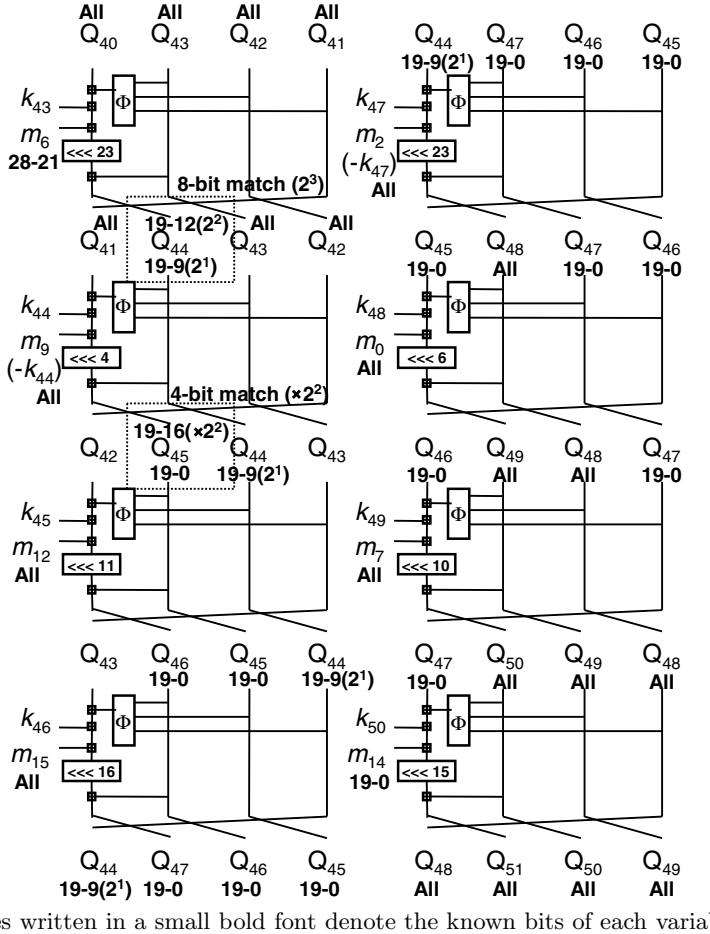
$$Q_{11} + \Phi_{14} + k_{14} + (m_{14} \wedge \texttt{0xfffff}) = ((-Q_{14}) \ggg s_{14}) \quad Pr. = 1, \quad (4)$$

$$((m_{14} + ((-Q_{14}) \ggg s_{14})) \lll s_{14}) + Q_{14} = (m_{14} \lll s_{14}) \; Pr. = 1 - 2^{-17}. \quad (5)$$

5.3 Details of Partial-Fixing for Skipping 8 Steps

As is explained in Section 4.3, meet-in-the-middle for skipping 8 steps will need to deal with unknown carried number behavior. The number of bits matched and number of unknown carried numbers depend on the number of rotations in each step. For the chunk we chose, we can apply 12-bit matching including 5 unknown carried numbers. We explain how the partial computation is performed step by step. The schematic explanation is in Figure 4. We use a notation $X^{b_2-b_1}$ to denote that values of bit positions b_1 to b_2 of a variable X are known.

Inverse computation for Steps 50-48. This is exactly the same as the partial-fixing technique used in Ref. [1]. In details, the equation for computing



Figures written in a small bold font denote the known bits of each variable.

Fig. 4. Partial-matching for 8 steps

Q_{47} in $R_{50}^{-1}(p_{51}, m_{50})$ is as follows.

$$Q_{47} = ((Q_{51}^{31-0} - Q_{50}^{31-0}) \ggg s_{50}) - \Phi_{50}(Q_{50}^{31-0}, Q_{49}^{31-0}, Q_{48}^{31-0}) - m_{\pi(50)}^{19-0} - k_{50}. \tag{6}$$

k_j is constant, hence k_j is known value. Because the lower 20 bits (positions 0 to 19) of $m_{14}(= m_{\pi(50)})$ are fixed and known, we can uniquely obtain the lower 20 bits of Q_{47} independently of the upper 12 bits of m_{14}. Similarly, the lower 20 bits of Q_{46} and Q_{45} can be uniquely computed as follows:

$$Q_{46} = ((Q_{50}^{31-0} - Q_{49}^{31-0}) \ggg s_{49}) - \Phi_{49}(Q_{49}^{31-0}, Q_{48}^{31-0}, Q_{47}^{19-0}) - m_{\pi(49)}^{31-0} - k_{49}, \tag{7}$$

$$Q_{45} = ((Q_{49}^{31-0} - Q_{48}^{31-0}) \ggg s_{48}) - \Phi_{48}(Q_{48}^{31-0}, Q_{47}^{19-0}, Q_{46}^{19-0}) - m_{\pi(48)}^{31-0} - k_{48}. \tag{8}$$

Inverse computation for Step 47. Equation for Q_{44} is as follows:

$$Q_{44} = ((Q_{48}^{31-0} - Q_{47}^{19-0}) \ggg s_{47}) - \Phi_{47}(Q_{47}^{19-0}, Q_{46}^{19-0}, Q_{45}^{19-0}) - m_{\pi(47)}^{31-0} - k_{47}. \tag{9}$$

We can uniquely compute the lower 20 bits of $Q_{48} - Q_{47}$. Let the value after the right rotation by $23(= s_{47})$ bits be u, and then, we uniquely obtain u^{28-9}. We can also compute the lower 20 bits of the output of Φ_{47}. Set the value of $m_2(= m_{\pi(47)})$ to $-k_{47}$ in advance. Then, the equation (9) becomes $u^{28-9} - \Phi_{47}^{19-0}$. By considering two possible carried number patterns from bit position 8 to 9, we can obtain two candidates of Q_{44}^{19-9}.

Forward computation for Step 43. $m_6(= m_{\pi(43)})$ in bit positions 21-28 are fixed. Then, the equation for Q_{44} in $R_{43}(p_{43}, m_{\pi(43)})$ is as follows.

$$Q_{44} = Q_{43}^{31-0} + (Q_{40}^{31-0} + \Phi_{43}(Q_{43}^{31-0}, Q_{42}^{31-0}, Q_{41}^{31-0}) + m_{\pi(43)}^{28-21} + k_{43}) \lll s_{43}. \tag{10}$$

By considering two possible carried number patterns from bit 20 to 21, we obtain two candidates of bit positions 21-28 of $m_6 + (Q_{40} + \Phi_{43} + k_{43})$. Let the value after the left rotation by $23(= s_{43})$ bits be v, and thus, we obtain two candidates of v^{19-12}. Finally, by considering two carried number patterns from bit 11 to 12 in the addition of Q_{43}, we obtain two candidates of Q_{44}^{19-12} for each v^{19-12}. (In total, we obtain 2^2 candidates of Q_{44}^{19-12} for each p_{43}.)

Forward computation for Step 44. The equation for Q_{45} is as follows.

$$Q_{45} = Q_{44}^{19-12} + (Q_{41}^{31-0} + \Phi_{44}(Q_{44}^{19-12}, Q_{43}^{31-0}, Q_{42}^{31-0}) + m_{\pi(44)}^{31-0} + k_{44}) \lll s_{44}. \tag{11}$$

We set $m_9(= m_{\pi(44)})$ to $-k_{44}$ to ignore the addition of these values. Bits 12-19 of Φ_{44} can be computed. Then, we obtain two candidates of $(Q_{41} + \Phi_{44})^{19-12}$. After the left rotation by $4(= s_{44})$ bits, known bits are moved to 16-23. Finally, after the addition of Q_{44}, we obtain two candidates of Q_{45}^{19-16} for each $(Q_{41} + \Phi_{44})^{19-12}$. (In total, we obtain 2^2 candidates of Q_{45}^{19-16} for each Q_{44}^{19-12}.)

As a result, by comparing forward and backward computation results, we can compare Q_{44}^{19-12} in total 8 bits with 3 unknown carried numbers and Q_{45}^{19-16} in total 4 bits with 2 unknown carried numbers. Therefore, our attack overall performs 12-bit match with 5 unknown carried numbers.

5.4 Attack Procedure

The attack procedure for a given hash value H_n is as follows:

1. Set chaining variables in the initial structure as shown in Figure 3.
2. Set m_2, m_9, m_{15}, and part of m_6 and m_{14} as shown in Figures 3 and 4. Set other message words to randomly chosen values but satisfy the padding.
3. For all possible values of bit positions 0-20 and 29-31 of m_6 and bit positions 0-4 and 17-31 of Q_{14}, in total 44 free-bits,
   (a) Compute Q_{11} by equation (3),
   (b) Compute $Q_{14}+m_6$ for efficient consistency check. Let this value be $C^{1\text{st}}$.
   (c) Do the following.

$$\begin{cases} p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 13, 12, \ldots, 0, \\ p_{64} \leftarrow H_n - p_0, & \\ p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 63, 62, \ldots, 51, \end{cases}$$

   (d) Compute $Q_{47}^{19-0}, Q_{46}^{19-0}$, and Q_{45}^{19-0} by equations (6), (7), and (8).
   (e) Compute two candidates of Q_{44}^{19-12} by equation (9).
   (f) Make a table of $(m_6, Q_{14}, C^{1\text{st}}, p_{51}, Q_{47}, Q_{46}, Q_{45}, Q_{44})$.
4. For all possible values of bit positions 20-31 of m_{14} and all bits of Q_{18}, in total 44 free-bits,
   (a) Compute Q_{15}, Q_{16}, and Q_{17} as shown in Figure 3.
   (b) Compute $((Q_{18} - Q_{17}) \ggg s_{17}) - \Phi_{17} - k_{17}$ for the efficient consistency check. Let this value be $C^{2\text{nd}}$.
   (c) Compute $p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)})$ for $j = 18, 19, \ldots, 42$.
   (d) i. Compute 2^2 candidates of Q_{44}^{19-12} for each p_{43} by equation (10), and Q_{45}^{19-16} for each Q_{44}^{19-12} by equation (11). In total, for each p_{43}, we obtain 2^4 candidates of $(Q_{44}^{19-12}, Q_{45}^{19-16})$.
   ii. Check whether bits 12-19 of Q_{44} and bits 16-19 Q_{45} in total 12 bits are matched with those in the table and $C^{2\text{nd}}$ is matched with $C^{1\text{st}}$ in the table.
   iii. If matched, compute $R_{43}(p_{43}, m_6)$ by corresponding m_6 and check whether bits 9-11 of Q_{44} are matched and the carried number assumption of Q_{44} is correct.
   iv. If matched, compute $R_{44}(p_{44}, m_9)$ and check whether bits 0-15 of Q_{45} are matched and the carried number assumption of Q_{45} is correct.
   v. Similarly, compute Q_{46} to Q_{51} and check the matching. If all bits are matched, the corresponding (p_0, M) is a pseudo-preimage.

5.5 Complexity Evaluation

Let the complexity of 1 step function be $\frac{1}{64}$ MD5 compression function.

Steps 1 and 2: Negligible.
Step 3a: The complexity is $2^{44} \cdot \frac{1}{64}$.
Step 3b: The complexity is much less than $2^{44} \cdot \frac{1}{64}$.
Step 3c: The complexity is $2^{44} \cdot \frac{27}{64}$.
Step 3d: The complexity is $2^{44} \cdot \frac{3}{64}$.
Steps 3e, 3f: The complexity is $2^{44} \cdot 2^1 \cdot \frac{1}{64}$ and provides 2^{45} items in the table.
Step 4a: The complexity is $2^{44} \cdot \frac{3}{64}$.
Step 4b: The complexity is $2^{44} \cdot \frac{1}{64}$.
Step 4c: The complexity is $2^{44} \cdot \frac{25}{64}$.
Step 4(d)i: The complexity is $2^{44} \cdot 2^2 \cdot \frac{1}{64} + 2^{44} \cdot 2^{2+2} \cdot \frac{1}{64}$, and provides 2^{48} candidates.
Step 4(d)ii: Comparison can be performed with negligible cost by the standard meet-in-the-middle method. The number of remaining pairs is $2^{49} (= 2^{45} \cdot 2^{48} \cdot 2^{-12} \cdot 2^{-32})$.
Step 4(d)iii: The complexity is $2^{43} (= 2^{49} \cdot \frac{1}{64})$. The number of remaining pairs is $2^{44} (= 2^{49} \cdot 2^{-3} \cdot 2^{-2})$.
Step 4(d)iv: The complexity is $2^{38} (= 2^{44} \cdot \frac{1}{64})$. The number of remaining pairs is $2^{26} (= 2^{44} \cdot 2^{-16} \cdot 2^{-2})$.
Step 4(d)v: The complexity is negligible compared to those of the other steps.

The sum of the above complexity is $2^{44} \cdot \frac{116}{64} \approx 2^{44.86}$. This means that we can obtain 2^{44} pairs where 12 bits are matched with a complexity of $2^{44.86}$. Therefore, by repeating the above procedure 2^{72} times, we expect to obtain a pseudo-preimage. Finally, the complexity of finding a pseudo-preimage of MD5 is $2^{116.86} (= 2^{44.86} \cdot 2^{72})$, and this is converted to a preimage attack with a complexity of $2^{123.43} \approx 2^{123.4}$ with the conversion algorithm explained in Section 3.1.

In the attack procedure, the dominant memory complexity is for Step 3f, which requires 2^{45} $(m_6, Q_{14}, C^{1\text{st}}, p_{51}, Q_{47}, Q_{46}, Q_{45}, Q_{44})$s to be stored. Therefore the memory complexity of our attack is at most $2^{45} \times 11$ words.

Remarks

Because the value of m_{14}, which is the lower 32-bits of the message length string, is not fixed in our attack, we cannot fix the length of preimage in advance. Therefore, when we convert pseudo-preimages to a preimage, the required message length is different for each pseudo-preimage. This problem is solved by using *expandable message* described in [7]. Note the cost for constructing an expandable message is negligible compared to the complexity of the preimage attack.

6 Conclusion

This paper shows a preimage attack on full MD5. Compared to the previous preimage attacks, we developed several new techniques: the initial structure,

which is a generalization of the previous local-collision technique, the cross absorption properties, the partial-fixing technique for unknown carried number behavior, and the efficient consistency check method for the initial structure. By combining these techniques, our attack with a complexity of $2^{116.9}$ finds a pseudo-preimage of full MD5, and with a complexity of $2^{123.4}$ finds a preimage of full MD5. The memory complexity of the attack is $2^{45} \times 11$ words.

References

1. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Workshop Records of SAC 2008, Sackville, Canada, pp. 82–98 (2008)
2. Aumasson, J.-P., Meier, W., Mendel, F.: Preimage attacks on 3-pass HAVAL and step-reduced MD5. In: Workshop Records of SAC 2008, Sackville, Canada, pp. 99–114 (2008) (ePrint version is avaliable at IACR Cryptology ePrint Archive: Report 2008/183), `http://eprint.iacr.org/2008/183.pdf`
3. De Cannière, C., Rechberger, C.: Preimages for reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008) (slides on preliminary results were appeared at ESC 2008 seminar), `http://wiki.uni.lu/esc/`
4. De, D., Kumarasubramanian, A., Venkatesan, R.: Inversion attacks on secure hash functions using SAT solvers. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 377–382. Springer, Heidelberg (2007)
5. den Boer, B., Bosselaers, A.: Collisions for the compression function of MD-5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
6. Dobbertin, H.: The status of MD5 after a recent attack. CryptoBytes The technical newsletter of RSA Laboratories, a division of RSA Data Security, Inc., 2(2) (Summer, 1996)
7. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
8. Klima, V.: Tunnels in hash functions: MD5 collisions within a minute. In: IACR Cryptology ePrint Archive: Report 2006/105 (2006), `http://eprint.iacr.org/2006/105.pdf`
9. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
11. Rivest, R.L.: Request for Comments 1321: The MD5 Message Digest Algorithm. The Internet Engineering Task Force (1992), `http://www.ietf.org/rfc/rfc1321.txt`
12. Sasaki, Y., Aoki, K.: A preimage attack for 52-steps HAS-160. In: Preproceedings of Information Security and Cryptology ICISC 2008 (2008)
13. Sasaki, Y., Aoki, K.: Preimage attacks on step-reduced MD5. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 282–296. Springer, Heidelberg (2008)

14. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J.P. (ed.) Advances in Cryptology - ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
15. U.S. Department of Commerce, National Institute of Standards and Technology. Federal Register, vol. 72(212) Friday, November 2, 2007/Notices, (2007) http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
16. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

Asymmetric Group Key Agreement

Qianhong Wu[1,2], Yi Mu[3], Willy Susilo[3], Bo Qin[1,4], and Josep Domingo-Ferrer[1]

[1] Universitat Rovira i Virgili, Dept. of Comp. Eng. and Maths
UNESCO Chair in Data Privacy, Tarragona, Catalonia
{qianhong.wu,bo.qin,josep.domingo}@urv.cat
[2] Key Lab. of Aerospace Information Security and Trusted Computing
Ministry of Education, School of Computer, Wuhan University, China
[3] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
{ymu,wsusilo}@uow.edu.au
[4] Dept. of Maths, School of Science, Xi'an University of Technology, China

Abstract. A group key agreement (GKA) protocol allows a set of users to establish a common secret via open networks. Observing that a major goal of GKAs for most applications is to establish a confidential channel among group members, we revisit the group key agreement definition and distinguish the conventional (*symmetric*) group key agreement from *asymmetric group key agreement* (ASGKA) protocols. Instead of a common secret key, only a shared encryption key is negotiated in an ASGKA protocol. This encryption key is accessible to attackers and corresponds to different decryption keys, each of which is only computable by one group member. We propose a generic construction of *one-round* ASGKAs based on a new primitive referred to as aggregatable signature-based broadcast (ASBB), in which the public key can be simultaneously used to verify signatures and encrypt messages while any signature can be used to decrypt ciphertexts under this public key. Using bilinear pairings, we realize an efficient ASBB scheme equipped with useful properties. Following the generic construction, we instantiate a one-round ASGKA protocol *tightly* reduced to the decision Bilinear Diffie-Hellman Exponentiation (BDHE) assumption in the standard model.

1 Introduction

Many complex cryptosystems rely on the existence of a confidential channel among the users. A major goal of key agreement protocols is to establish such a channel for two or more users. Since the inception of the Diffie-Hellman protocol [12] in 1976, it has been an elusive open problem to construct a *one-round* group key agreement protocol *from scratch*. A *round* means that each party sends one message and can broadcast simultaneously. A key agreement protocol is said to be *from scratch* if each participant does not hold any secret values prior to the execution of the protocol. Each type of long-term-key free protocols can only provide security against passive attackers, but they are the basis to build

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 153–170, 2009.

advanced protocols against more powerful attackers. We concentrate on new key agreement protocols from scratch in this report.

1.1 Our Contribution

This paper investigates a close variation of the above mentioned problem of one-round group key agreement protocols and focuses on *"how to establish a confidential channel from scratch for multiple parties in one round"*. We provide a short overview of some new ideas to solve this variation.

Asymmetric GKA. Observe that a major goal of GKAs for most applications is to establish a confidential broadcast channel among the group. We investigate the potentiality to establish this channel in an asymmetric manner in the sense that the group members merely negotiate a common encryption key (accessible to attackers) but hold respective secret decryption keys. We introduce a new class of GKA protocols which we name *asymmetric group key agreements* (ASGKAs), in contrast to the conventional GKAs. A trivial solution is for each member to publish a public key and withhold the respective secret key, so that the final ciphertext is built as a concatenation of the underlying individual ones. However, this trivial solution is highly inefficient: the ciphertext increases linearly with the group size; furthermore, the sender has to keep all the public keys of the group members and separately encrypt for each member. We are interested in nontrivial solutions that do not suffer from these limitations.

Aggregatable signature-based broadcast (ASBB). Our proposals rely on a new notion named *aggregatable signature-based broadcast*. In an ASBB scheme, the public key can be simultaneously used to verify signatures and encrypt messages, and any valid signature can be used to decrypt ciphertexts under this public key; furthermore, an ASBB scheme satisfies the *key-homomorphic property* and the *aggregatability* property. The key-homomorphic property means that the combination of signatures on the *same message* produces a valid signature of this message under the combination of the corresponding public keys. As a consequence, the combined signature can be used as a decryption key of the new ASBB instance. Aggregatability states that the combination of secure ASBB instances produces a new secure ASBB instance.

Non-trivial one-round ASGKA. We propose a non-trivial one-round ASGKA scheme. Our idea is to generate the public key of an ASBB scheme in a distributed manner, such that only each member can obtain a signature under this public key. These signatures can be used as their respective decryption keys and a confidential channel among the group is established. We build an efficient ASBB scheme from bilinear pairings and prove its security under the decision n-Bilinear Diffie-Hellman exponentiation (n-BDHE) assumption with the help of a random oracle. By following the generic construction and exploiting the randomness in the setup stage, we instantiate a one-round ASGKA protocol and tightly reduce its security to the decision n-BDHE assumption in the standard model (without using random oracles). The proposed one-round ASGKA protocol achieves the confidential channel of a one-round conventional GKA protocol.

Also, our ASGKA proposal has additional advantages, *e.g.*, serving as a public key based broadcast scheme without requiring a dealer.

1.2 One-Round Group Key Agreement

A key agreement protocol enables two or more users within an open network to create a common secret key. In what follows we review round-efficient key agreement protocols, especially, one-round protocols. These protocols are unauthenticated and only secure against passive attacks but they are the basis on which protocols with active security can be built. To date, only very few one-round key agreement protocols (excluding their variations) have been found, *i.e.*, the two-party Diffie-Hellman [12] and the tripartite Joux [19] protocols.

The basic Diffie-Hellman protocol [12] is a one-round two-party protocol. After each party publishes one element in a finite cyclic group, two parties can establish a common secret key. If one party fixes its published element as its public key and the other party generates its element dynamically for each session, the Diffie-Hellman protocol implies a public key cryptosystem, *i.e.*, the well-known ElGamal cryptosystem [13].

The Joux protocol [19] is a one-round tripartite protocol. Similarly to the Diffie-Hellman protocol, by fixing two members' published strings as their public keys, the Joux protocol implies a mini broadcast cryptosystem without a trusted dealer in which only these two members can decrypt the messages. Nevertheless, the Joux protocol is limited to three parties. To date, it remains unknown whether it can be extended to more than three parties without introducing additional rounds.

For more than three parties, Boneh and Silverberg proposed a one-round $(n+1)$-party protocol [6] but their protocol relies on n-linear pairings which by itself is also an open problem and no construction has been found so far. Assuming that there exist n-linear pairing maps, their protocol implies a broadcast cryptosystem [4,14] for n users by fixing n users' published strings as their public keys. Similarly to the Joux protocol, the Boneh-Silverberg protocol is limited to $n+1$ parties. It seems difficult to extend it to more than $n+1$ parties without additional rounds even if the existence of efficient n-linear pairings is assumed.

Burmester and Desmedt [11] extended the Diffie-Hellman protocol to n parties. Their protocol requires two rounds and is the most efficient existing GKA protocol in round efficiency without constraints on n. Some papers (*e.g.*, [7,22]) can achieve authenticated GKA in one-round. But these GKA protocols are based on public key encryption and differ from the above ones in that they are *not from scratch*. The protocol in [7] is PKI-based and only one party uses its public key for encryption purpose. Since setting up a PKI may be regarded as a one-round protocol, such protocols can fairly be compared with a two-round protocol from scratch. It is an open question whether our one-round protocol can be used to build a two-round authenticated GKA from scratch.

However, as remarked by Joux [19], in some cases the two rounds in key agreement protocols can be somewhat cumbersome, and a single pass would be much more preferable. For instance, exchanging an email message key among a

group of users with a two-round key agreement protocol would require all of them to be connected concurrently. Another scenario is a group of friends wishing to share their private files via the insecure Internet; doing so with a two-round key agreement protocol would require all of them to be online at the same time. In practice, it is difficult for a group of distributed users to be online concurrently (especially if they live in different time zones). In these scenarios, the bandwidth is not a problem but the round efficiency is critical. In addition, the bandwidth overhead can be mitigated by the hardware development but the round overhead can only be addressed by more efficient protocols.

Unauthenticated GKA protocols can only be secure against passive attackers. Hence, it is necessary to improve these basic protocols to meet active security for practical applications. There are lots of studies following this research line. In [1] Bellare *et al.* showed a compiler which converts unauthenticated protocols into authenticated ones in the two-party setting; extending [1] to the group setting is possible but inefficient. In [20], Katz and Yung proposed an aggregatable compiler which transforms any secure GKA protocol against passive attackers into a secure authenticated GKA protocol against active attackers. The compiler preserves forward security of the original protocol. In [21], Kudla and Paterson considered modular proof techniques for authenticated key agreement protocols which are not designed in a modular way but which one nevertheless wishes to prove secure. Different key agreement protocols secure against active attackers may be constructed from authentication techniques and a library of basic protocols including our protocols in this paper.

1.3 Motivating Applications

By way of motivation, we illustrate some interesting applications and advantages of the new notion of ASGKA as well as our one-round instantiation.

Application scenarios. Our ASGKA protocol suits broadcast applications to which regular broadcast schemes and GKA protocols are difficult to apply. We have in mind applications where it is hard to find a trusted party to serve as a dealer in a regular broadcast scheme, and it is inconvenient to require all the parties to stay online concurrently to implement a (two-round) regular GKA protocol. Such applications include broadcast to *ad hoc* groups, off-line file sharing via internet, secure group chat, group purchase of encrypted content with identity privacy (*i.e.*, where the seller cannot identify the members of a group purchase) and so on. These applications deal with not very large self-organized groups which live a period of time after being established. Hence, our ASGKA fills the application gap left by regular GKA or broadcast schemes.

Comparison with the trivial solution. As mentioned in Section 1.1, the trivial solution suffers from linear complexity in the ciphertext size, keys storage requirement and encryption overhead. In our scheme, the ciphertext, the negotiated public key and all decryption keys are of constant size. Although in our proposal each member's published string is linear in the group size in negotiation, no one needs to keep any bit of them after executing the protocol. Hence, the storage requirement is small in our scheme. The encryption operation in our

scheme is also efficient, *i.e.*, three exponentiations. Due to the heavy communication overhead in key establishment, our scheme does not improve on the trivial solution for one-time group applications[1]. However, in practice, once a group is established, it is likely to live for a period of time, as discussed above, in which case the communication overhead incurred by our protocol can be amortized over the group lifetime.

Public key based broadcast without a dealer. For our ASGKA proposal, if each member is allowed to register to a certificate authority its published string as its public key, then anyone knowing the public keys of all members can send encrypted messages to the group and only the group members can decrypt the message. Here, the ciphertexs and the secret key of each member are constant and independent of the group scale. This implies that our ASGKA protocol can be used as an efficient public key based dealer-free broadcast scheme which, to the best of our knowledge, is not found in the public literature. However the existing broadcast schemes in the literature do require such a privileged dealer and confidential channels from the dealer to each subscribers before implementing the broadcast system, which is undesirable in some applications.

Key self-confirmation. After a GKA protocol is executed, the agreed keys may become invalid for various reasons, for instance, inside attackers or communication errors. Some proposals (e.g., [10, 20]) suggest a simple method to complete the key confirmation. They assume that the group members have agreed on a public message in advance. After running the basic protocols, each member encrypts this message with its derived session key and broadcasts the ciphertext. Then the members can verify whether they share the same session key (the existing GKAs are symmetric) by checking the consistency of the ciphertexts. This method may not work if there exist inside attackers (a major goal of the key confirmation is to prevent inside attacks). Such attackers can invalidate a correct session key by simply sending a random string in the last round. This flaw can be fixed by letting the group members prove that they have behaved honestly, but such fixes seem expensive in practice. For our asymmetric GKA protocol, the key confirmation is simple and requires no additional rounds if the protocol has been correctly executed. Group members can choose a random message and encrypt it using their derived encryption keys. Then they can decrypt using their derived decryption keys. If the decryption outputs a correct plaintext, the corresponding member concludes that the ASGKA protocol has been correctly run; otherwise, the member raises a public alarm. Hence, if an ASGKA protocol has been correctly executed, each member can verify this fact locally without communicating with others and no additional rounds are required.

Identification of disruptive principals in GKA protocols. In existing GKA protocols, it is not a trivial job to identify malicious principals aiming to disrupt the protocol even if some of these protocols are authenticated and proven secure against active attackers. Indeed, authentication in a GKA protocol only prevents such attacks by outside adversaries. A disruptive principal can just

[1] Those applications are about fully dynamic group broadcast without a dealer and could be interesting for future research.

broadcast some authenticated random strings during the protocol execution to paralyze the protocol. To thwart such attackers, group members may be required to prove that they have behaved correctly in a zero-knowledge manner. This modification can work (to identify the disruptive principals) but it introduces a substantial cost. In our ASGKA scheme, since the transcripts from participants are indeed signatures under their respective temporary public keys, anyone who did not honestly behave can be identified and expelled if any group member raises an alarm after key confirmation.

1.4 Paper Organization

In Section 2, we revisit the notion of group key agreement. Section 3 presents a generic ASGKA construction from ASBBs with key homomorphic property and aggregatability. An ASBB scheme is efficiently realized in Section 4 and the one-round ASGKA protocols naturally follow from the generic formula. Section 5 is a conclusion.

2 Group Key Agreement Revisited

In the conventional GKA definition, a group of members interact to establish a common secret key within an open network. One may notice that a major goal of GKAs is to establish a confidential broadcast channel among the group. In such a broadcast context from GKA protocols, there is no need for a dealer to distribute decryption keys, but only group members can broadcast to other group members[2]. However, in practice the senders can be potentially anyone, as the case of asymmetric encryption. Hence, we are motivated to find alternatives to achieve the final goal of conventional GKAs. For instance, assume that there exists a public key based broadcast cryptosystem in which the single public key corresponds to numerous secret decryption keys; if the group members can jointly generate such a cryptosystem and share the public key while only the group members can extract a secret key during the joint computation, then any message encrypted under this public key can only be decrypted by the group members and a confidential channel is built among them.

2.1 Protocol Variables and Partner Relationship

The following revisited GKA definition derives from the widely-accepted ones due to [7,8,9,10,20]. Similarly to [20], we assume for simplicity a fixed, polynomial-size set $\mathbb{P} = \{\mathcal{U}_1, \cdots, \mathcal{U}_\ell\}$ of potential players. Any subset of the potential players may decide at any point to establish a confidential channel among them, but we assume that these subsets are the same during a run of the protocol and concentrate on static groups.

[2] In some applications, this feature can be an advantage, *e.g.*, intra-group authentications. In our ASGKA protocol, intra-group authentication can be easily realized since the underlying primitive can be used as a signature scheme.

Whenever group membership changes, a new group $\mathbb{P}_v = \{\mathcal{U}_1, \cdots, \mathcal{U}_n\}$ is formed and its members can establish a confidential channel through an instance performing a group key agreement protocol Σ: the index v increases whenever the membership changes and $\mathbb{P}_0$ denotes the initial group. $\Pi_{\mathcal{U}_i}^{\imath_i}$ denotes an instance $\imath_i$ of a group member $\mathcal{U}_i$. An instance $\Pi_{\mathcal{U}_i}^{\imath_i}$ has a unique session identifier $\mathtt{Sid}_{\mathcal{U}_i}^{\imath_i}$ and a partner identifier $\mathtt{Pid}_{\mathcal{U}_i}^{\imath_i}$. After the GKA protocol Σ has been terminated successfully, $\Pi_{\mathcal{U}_i}^{\imath_i}$ has a unique decryption key identifier $\mathtt{Dkid}_{\mathcal{U}_i}^{\imath_i}$ corresponding to a decryption key $dk_{\mathcal{U}_i}^{\imath_i}$, and a unique encryption key identifier $\mathtt{Ekid}_{\mathcal{U}_i}^{\imath_i}$ corresponding to an encryption key $ek_{\mathcal{U}_i}^{\imath_i}$, and a freshness identifier $\mathtt{Fid}_{\mathcal{U}_i}^{\imath_i}$ representing whether $dk_{\mathcal{U}_i}^{\imath_i}$ is compromised. If not, $\mathtt{Fid}_{\mathcal{U}_i}^{\imath_i} = 1$; else $\mathtt{Fid}_{\mathcal{U}_i}^{\imath_i} = 0$. Finally, the partner identifier $\mathtt{Pid}_{\mathcal{U}_i}^{\imath_i}$ corresponds to a set of group members $\mathbb{P}_{\mathcal{U}_i}^{\imath_i} = \mathbb{P}_v \setminus \{\mathcal{U}_i\}$.

Definition 1 (Successful termination of GKA). *We say that a GKA protocol Σ has been successfully terminated in the instance $\Pi_{\mathcal{U}_i}^{\imath_i}$ if for $1 \leq k \neq i \leq n$, (1) each $\mathcal{U}_k$ of $\mathbb{P}_{\mathcal{U}_i}^{\imath_i}$ has instance $\Pi_{\mathcal{U}_k}^{\imath_k}$ containing $\{\mathtt{Sid}_{\mathcal{U}_k}^{\imath_k}, \mathtt{Pid}_{\mathcal{U}_k}^{\imath_k}, \mathtt{Dkid}_{\mathcal{U}_k}^{\imath_k}, \mathtt{Ekid}_{\mathcal{U}_k}^{\imath_k}\}$; (2) $\mathtt{Sid}_{\mathcal{U}_k}^{\imath_k} = \mathtt{Sid}_{\mathcal{U}_i}^{\imath_i}$; (3) $\mathbb{P}_{\mathcal{U}_k}^{\imath_k} = \mathbb{P}_v \setminus \{\mathcal{U}_k\}$. In this case, we state that the instances $\Pi_{\mathcal{U}_i}^{\imath_i}$ and $\Pi_{\mathcal{U}_k}^{\imath_k}$ are partnered.*

2.2 Adversarial Model

In the real world, a protocol determines how principals behave in response to signals from their environment. In the model, these signals are sent by the adversary $\mathcal{A}$. For simplicity, only passive adversaries are considered in the definitions. A passive adversary is assumed to merely eavesdrop all communication in the network. An adversary $\mathcal{A}$'s interaction with the principals in the network (more specifically, with the various instances) is modeled by the following oracles:

- $\mathtt{Parameter}(1^\lambda)$: On $\mathcal{A}$'s query λ, respond with common parameters denoted by π, including two polynomial time algorithms $\mathcal{E}(\cdot,\cdot)$ and $\mathcal{D}(\cdot,\cdot)$.
- $\mathtt{Setup}(\mathbb{P}_0)$: On $\mathcal{A}$'s query $\mathbb{P}_0$, start the protocol Σ and output the initial group $\mathbb{P}_0 = \{\mathcal{U}_1, \cdots, \mathcal{U}_\ell\}$. For $1 \leq k \leq \ell$, initialize $\mathtt{Sid}_{\mathcal{U}_k}^{\imath_k} \leftarrow 0, \mathtt{Pid}_{\mathcal{U}_k}^{\imath_k} \leftarrow \varnothing, \mathtt{Dkid}_{\mathcal{U}_k}^{\imath_k} \leftarrow NULL, \mathtt{Ekid}_{\mathcal{U}_k}^{\imath_k} \leftarrow NULL, \mathtt{Fid}_{\mathcal{U}_k}^{\imath_k} \leftarrow 1, S \leftarrow 0$.
- $\mathtt{Execute}(\mathcal{U}_1, \cdots, \mathcal{U}_n)$: Execute the protocol between unused instances of players $\{\mathcal{U}_1, \cdots, \mathcal{U}_n\} = \mathbb{P}_v \subseteq \mathbb{P}_0$ and output the transcript of the execution. Here, v changes whenever the group changes and hence is the group sequence number. The number of group members and their identities are chosen by the adversary. For $1 \leq k \leq n$, update $\mathtt{Sid}_{\mathcal{U}_k}^{\imath_k} \leftarrow \mathtt{Sid}_{\mathcal{U}_k}^{\imath_k} + 1, \mathtt{Pid}_{\mathcal{U}_k}^{\imath_k} \leftarrow \mathbb{P}_v \setminus \{\mathcal{U}_k\}$, $\mathtt{Dkid}_{\mathcal{U}_k}^{\imath_k} \leftarrow dk_{\mathcal{U}_k}^{\imath_k}, \mathtt{Ekid}_{\mathcal{U}_k}^{\imath_k} \leftarrow ek_{\mathcal{U}_k}^{\imath_k}, S \leftarrow S + 1$. S is the session sequence number to record the running times of $\mathtt{Execute}$.
- $\mathtt{Ek\text{-}Reveal}(\Pi_{\mathcal{U}_i}^{\imath_i})$: Output $\mathtt{Ekid}_{\mathcal{U}_i}^{\imath_i}$.
- $\mathtt{Dk\text{-}Reveal}(\Pi_{\mathcal{U}_i}^{\imath_i})$: Output $\mathtt{Dkid}_{\mathcal{U}_i}^{\imath_i}$. Update $\mathtt{Fid}_{\mathcal{U}_i}^{\imath_i} \leftarrow 0$. We allow the encryption key to be different from the decryption key and hence the $\mathtt{Ek\text{-}Reveal}$ oracle and the $\mathtt{Dk\text{-}Reveal}$ oracle are distinguished.
- $\mathtt{Test}(\Pi_{\mathcal{U}_i}^{\imath_i}, m_0, m_1)$: This query is used to define the advantage of an adversary $\mathcal{A}$. $\mathcal{A}$ executes this query on a *fresh instance* $\Pi_{\mathcal{U}_i}^{\imath_i}$ (see Definition 4 below) at any time, but only once (other queries have no restriction). When $\mathcal{A}$ asks this query, it receives a challenge ciphertext $c^* = \mathcal{E}(m_\rho, ek_{\mathcal{U}_i}^{\imath_i})$, where ρ is the result of a coin flip. Finally, $\mathcal{A}$ outputs a bit ρ'.

2.3 Properties of GKA

A major goal of GKA protocols is to establish a confidential channel for group members. We use this goal to define the *correctness* of a GKA protocol while use the approaches to the goal to classify GKA protocols.

Definition 2 (Correctness). *We say a GKA protocol Σ is correct if, whenever it has been successfully terminated, for any instance $\Pi_{\mathcal{U}_i}^{\imath_i}$ and any of its partners $\Pi_{\mathcal{U}_k}^{\imath_k}$ and any message m in the message space of $\mathcal{E}(\cdot,\cdot)$, it holds that $\mathcal{D}(\mathcal{E}(m, ek_{\mathcal{U}_i}^{\imath_i}), dk_{\mathcal{U}_k}^{\imath_k}) = m$ and $\mathcal{D}(\mathcal{E}(m, ek_{\mathcal{U}_k}^{\imath_k}), dk_{\mathcal{U}_i}^{\imath_i}) = m$.*

Definition 3 (Asymmetric Group Key Agreement). *A GKA protocol Σ is said to be symmetric if after being successfully terminated, it holds that $dk_{\mathcal{U}_i}^{\imath_i} = dk_{\mathcal{U}_k}^{\imath_k} = sk$. Else, Σ is said to be an asymmetric group key agreement protocol.*

A GKA protocol is nontrivial if $|\mathcal{E}(m, ek_{\mathcal{U}_i}^{\imath_i})|$ and $|\mathcal{E}(m, ek_{\mathcal{U}_k}^{\imath_k})|$ are independent of n, where $|s|$ denotes the binary length of string s. This restraint rules out trivial protocols where each member just publishes a public key and keeps its secret key for an agreed public key cryptosystem. The conventional GKA protocols are all symmetric. An ASGKA protocol allows different members to have different decryption keys. However, it does not state whether the inside members can use their secret decryption keys to (collusively) compute a new (equivalent) decryption key. This issue of traitor traceability is beyond our scope.

We first define the notion of *freshness* as the precondition to define the secrecy of GKA protocols, in which the corruption query `Dk-Reveal` models insider attacks and captures forward security of GKAs.

Definition 4 (Freshness). *An instance $\Pi_{\mathcal{U}_i}^{\imath_i}$ is fresh if before the adversary answers the `Test` oracle, neither the instance $\Pi_{\mathcal{U}_i}^{\imath_i}$ nor anyone of its partnered instances has received a* `Dk-Reveal` *query.*

Let us proceed to the main security notion of GKAs against passive attackers. In the conventional GKA definitions, the group members finally share a secret key. Accordingly, the security is defined by the indistinguishability of the shared key from a random string. In our definition we allow the group members to have different decryption keys. Hence, we define the security of GKAs by the security of the final confidential channel, *i.e.*, the indistinguishability of messages transferred via this channel.

Definition 5 (Secrecy of GKA). *Let Σ be a GKA protocol and $\mathcal{A}$ be a passive adversary against Σ. When $\mathcal{A}$ asks a query $\texttt{Test}(\Pi_{\mathcal{U}_i}^{\imath_i}, m_0, m_1)$ to a fresh instance $\Pi_{\mathcal{U}_i}^{\imath_i}$ in Σ, $\mathcal{A}$ receives a challenge ciphertext $c^* = \mathcal{E}(m_\rho, ek_{\mathcal{U}_i}^{\imath_i})$, where ρ is the result of a coin flip. Finally, $\mathcal{A}$ outputs a bit ρ'. The advantage of $\mathcal{A}$ in the above secrecy game is defined as $Adv_{\mathcal{A},\Sigma}^{GKA} = |\Pr[\rho' = \rho] - \frac{1}{2}|$. Σ is said to be secure for static groups if $\mathcal{A}$ is allowed to query all the oracles and Adv_{Σ}^{GKA} is negligible.*

3 Generic Construction of One-Round ASGKA

In this section, we propose a secure generic construction of one-round ASGKA protocols for static groups. To achieve this goal, we present a primitive referred to as *aggregatable signature-based broadcast* (ASBB). It has the *key-homomorphic property* and *aggregatability* which are crucial for our generic one-round ASGKA protocol.

3.1 Aggregatable Signature-Based Broadcast

An ASBB scheme can be used simultaneously as a signature scheme and a broadcast scheme. It exploits the duality between decryption keys and signatures which has been noticed in ID-based encryption [3] and certificate-based encryption [17], but the encryption in ASBB does not take input the receivers' identities.

The security of an ASBB scheme incorporates the standard notion of security for a signature scheme, *i.e.*, existential unforgeability under a chosen message attack (EUF-CMA) [18] and the security as an encryption scheme.

Definition 6 (Aggregatable signature-based broadcast). *An ASBB scheme consists of six polynomial-time algorithms:*

- $\pi \leftarrow \texttt{ParaGen}(1^\lambda)$*: On input a security parameter* λ*, output the public parameters* π*.*
- $(pk, sk) \leftarrow \texttt{KeyGen}(\pi)$*: On input* π*, output a public/secret key pair* (pk, sk)*.*
- $\sigma \leftarrow \texttt{Sign}(pk, sk, s)$*: On input the key pair* (pk, sk) *and any string* s*, output a signature* σ*.*
- $0/1 \leftarrow \texttt{Verify}(pk, s, \sigma)$*: On input the public* pk *and a signature* σ *of string* s*, output 0 or 1.*
- $c \leftarrow \texttt{Encrypt}(pk, m)$*: On input a public key* pk *and a plaintext* m*, output a ciphertext* c*.*
- $m \leftarrow \texttt{Decrypt}(pk, s, \sigma, c)$*: On input the public key* pk*, any valid string-signature* (s, σ) *and a ciphertext* c*, output the plaintext* m*.*

Given the system parameters π*, a public key* pk *and a challenge ciphertext* $c = \texttt{Encrypt}(pk, m_\rho)$ *for* m_0 *and* m_1 *chosen by an attacker* $\mathcal{A}$*, where* ρ *is the challenger's random coin flip, the attacker wins if it outputs a guess bit* $\rho' = \rho$*. An ASBB scheme is said to be semantically indistinguishable against chosen plaintext attacks (Ind-CPA) if, for any polynomial-time attacker* $\mathcal{A}$*,* $|\Pr[\rho = \rho'] - \frac{1}{2}|$ *is negligible in* λ*. An ASBB scheme is said to be EUF-CMA-Ind-CPA secure if the underlying signature is EUF-CMA secure and the encryption is Ind-CPA secure.*

An ASBB scheme has a key-homomorphic property, as mentioned in Section 1.1. This means that, given two signatures on the same message under two public keys, one can efficiently produce a signature of the same message under a new public key derived from the original two public keys.

Definition 7 (Key homomorphism). *An ASBB scheme is said to be key homomorphic if, for any two public/secret key pairs* $(pk_1, sk_1), (pk_2, sk_2) \leftarrow \mathtt{KeyGen}(\pi)$ *and any message string* s, $\sigma_1 = \mathtt{Sign}(pk_1, sk_1, s)$, $\sigma_2 = \mathtt{Sign}(pk_2, sk_2, s)$, *it holds that (1)* $\mathtt{Verify}(pk_1 \otimes pk_2, s, \sigma_1 \odot \sigma_2) = 1$ *and (2)* $\mathtt{Decrypt}(pk_1 \otimes pk_2, s, \sigma_1 \odot \sigma_2, c) = m$ *for any plaintext* m *such that* $c \leftarrow \mathtt{Encrypt}(pk_1 \otimes pk_2, m)$, *where* $\otimes : \Gamma \times \Gamma \rightarrow \Gamma$ *and* $\odot : \Omega \times \Omega \rightarrow \Omega$ *are two efficient operations in the public key space* Γ *and the signature space* Ω, *respectively.*

For key homomorphic ASBB schemes sharing system parameters, the combination of signatures of the same string under different public keys is also a valid signature of this string under the combination of given public keys. Note that this property does not contradict the standard EUF-CMA security of signatures, since in a EUF-CMA game, the public key is provided by the challenger while the public key derived from combination is generated by the attacker in the key-homomorphic notion.

Let us consider this problem further. Since the combination of given public keys yields the public key of a new ASBB instance, we naturally question the security of the resulting ASBB instance such as the EUF-CMA security, Ind-CPA security or other properties that are potentially useful. For our focus of one-round GKAs, we study the Ind-CPA security of the resulting system and introduce another important property, *i.e.*, aggregatability, in an ASBB scheme.

Definition 8 (Aggregatability). *The aggregatability of an ASBB scheme is defined by the following game between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{CH}$*:*

- **Setup:** $\mathcal{A}$ *initializes the game with an integer* n. $\mathcal{CH}$ *replies with* $(\pi, pk_1, \cdots, pk_n)$ *which are the system parameters and* n *independent public keys of the key-homomorphic ASBB scheme.*
- **Education:** *For* $1 \leq j \neq i \leq n$, *the adversary* $\mathcal{A}$ *can adaptively choose any string* $s_j \in \{0,1\}^*$ *to query* $\mathcal{CH}$ *for a valid signature* $\sigma_i(s_j)$ *under* pk_i. s_j *is determined by the attacker but the limit is that* $s_i \neq s_j$ *if* $i \neq j$. *In other words, the attacker can freely decide* n *messages* $s_j (1 \leq j \leq n)$ *but it cannot query* s_j *to* PK_j, *although the queries to* PK_i *are allowable.*
- **Challenge:** $\mathcal{CH}$ *and* $\mathcal{A}$ *run a standard Ind-CPA game under the combined public key* $pk = pk_1 \otimes \cdots \otimes pk_n$. $\mathcal{A}$ *wins if* $\mathcal{A}$ *outputs a correct guess bit. Denote* $\mathcal{A}$*'s advantage by* $Adv_{\mathcal{A}} = |\Pr[win] - \frac{1}{2}|$.

An ASBB scheme is said to be (τ, ε, n)*-aggregatable against adaptively chosen message attacks if no* τ*-time algorithm* $\mathcal{A}$ *has advantage* $Adv_{\mathcal{A}} \geq \varepsilon$ *in the above aggregatability game. An ASBB scheme is said to be* (τ, ε, n)*-aggregatable against non-adaptively chosen message attacks if the adversary is required to provide* s_j *for* $j = 1, \cdots, n$ *after the* Setup *stage and no* τ*-time algorithm* $\mathcal{A}$ *has advantage* $Adv_{\mathcal{A}} \geq \varepsilon$ *in the corresponding non-adaptive aggregatability game.*

The aggregatability against non-adaptively chosen message attacks is sufficient for our purpose of one-round GKAs. Note that, in the above aggregatability definition, the EUF-CMA security is implicitly used. This is because if the underlying ASBB is not EUF-CMA secure, the attacker may successfully forge signatures of some message s' under all the public keys after the Education stage.

Due to the key-homomorphic property, the attacker can obtain a signature σ' of s' under pk and σ' is also a valid decryption key. As a result, the attacker can decrypt normally and the aggregatability cannot hold.

3.2 A Generic Construction of One-Round ASGKA Protocols

Based on ASBBs, we present a generic construction of ASGKA protocols for static groups. The construction is illustrated in Matrix (1), where $\Downarrow/\downarrow$, $\prec$: and `TPK` represent public/private computation, broadcast operation and temporary public key, respectively.

$$\begin{pmatrix} & & \mathcal{U}_1\texttt{knows} & \mathcal{U}_2\texttt{knows} & \mathcal{U}_3\texttt{knows} & \cdots & \mathcal{U}_n\texttt{knows} & \texttt{TPK} \\ \mathcal{U}_1 & \prec: & \bigcirc & \sigma_1(ID_2) & \sigma_1(ID_3) & \cdots & \sigma_1(ID_n) & pk_1 \\ \mathcal{U}_2 & \prec: & \sigma_2(ID_1) & \bigcirc & \sigma_2(ID_3) & \cdots & \sigma_2(ID_n) & pk_2 \\ \mathcal{U}_3 & \prec: & \sigma_3(ID_1) & \sigma_3(ID_2) & \bigcirc & \cdots & \sigma_3(ID_n) & pk_3 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathcal{U}_n & \prec: & \sigma_n(ID_1) & \sigma_n(ID_2) & \sigma_n(ID_3) & \cdots & \bigcirc & pk_n \\ & & \downarrow & \downarrow & \downarrow & \cdots & \downarrow & \Downarrow \\ \texttt{example:} & & dk_1 & dk_2 & dk_3 & \cdots & dk_n & pk \end{pmatrix} \tag{1}$$

Protocol I: One-Round ASGKA Protocol

- `Public Parameter Generation.` The public parameters are the description π of an ASBB scheme.
- `Group Setup.` Decide the group of the players $\mathbb{P}=\{\mathcal{U}_1,\cdots,\mathcal{U}_n\}$. Let $ID_1,\cdots,ID_n$ be the respective identities.
- `Group Key Agreement.` $\mathcal{U}_i$ randomly generates a public key pk_i and broadcasts the corresponding row in Matrix (1):

$$\sigma_i(ID_1),\cdots,\sigma_i(ID_{i-1}),\sigma_i(ID_{i+1}),\cdots,\sigma_i(ID_n),pk_i.$$

  Here, $\sigma_i(ID_j)$ is a signature of ID_j corresponding to the public key pk_i. $\bigcirc$ *represents that* $\sigma_i(ID_i)$ *is not published for* $1 \le i \le n$.
- `Group Encryption Key Derivation.` The group encryption key is:

$$pk = \bigotimes_{j=1}^{n} pk_j.$$

- `Decryption Key Derivation.` Player $\mathcal{U}_i$ can calculate its secret decryption key dk_i from the corresponding column of Matrix (1):

$$dk_i = \sigma_i(ID_i) \overset{n,j\neq i}{\underset{j=1}{\odot}} \sigma_j(s_i) = \overset{n}{\underset{j=1}{\odot}} \sigma_j(ID_i).$$

Note that an attacker cannot compute dk_i since $\sigma_i(ID_i)$ is unpublished. The last row is the output of a successful run of an ASGKA instance.

– `Encryption/Decryption`. Since π is an ASBB scheme, dk_i is a signature of s_i under the new public key pk. Hence, dk_i is also a decryption key corresponding to the new public key pk and the `Encryption/Decryption` algorithms can be trivially realized in π.

We give the proof of the following theorem in [23].

Theorem 1. *The proposed generic one-round ASGKA protocol for static groups is secure if the underlying key homomorphic ASBB is EUF-CMA-Ind-CPA secure and aggregatable against non-adaptive chosen message attacks.*

Although we only consider the CPA security in our ASGKA protocol, by noting that the resulting output of the protocol can be viewed as a public key based broadcast system, it is possible to achieve security against chosen ciphertext attacks (CCA). In fact, some generic approaches that convert a CPA secure encryption scheme into a CCA secure one can apply to our scheme, similarly to the Fujisaki-Okamoto conversion [15]. Since our focus is one-round ASGKAs with basic security, we will not explore such improvements here.

4 The Proposal

From the generic construction, to realize one-round ASGKA protocols, one needs only to implement a secure ASBB scheme. We construct an ASBB scheme secure in the random oracle model using available bilinear pairing techniques.

4.1 An Efficient ASBB Scheme

The scheme is realized in bilinear pairing groups [5,16]. Let `PairGen` be an algorithm that, on input a security parameter 1^λ, outputs a tuple $\Upsilon = (p, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ have the same prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficient non-degenerate bilinear map such that $e(g, g) \neq 1$ for any generator g of $\mathbb{G}$, and for all $u, v \in \mathbb{Z}$, it holds that $e(g^u, g^v) = e(g, g)^{uv}$.

– **Public parameters:** Let $\Upsilon = (p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathtt{PairGen}(1^\lambda)$, $\mathbb{G} = \langle g \rangle$. Let $H : \{0,1\}^* \to \mathbb{G}$ be a cryptographic hash function. The system parameters are $\pi = (\Upsilon, g, H)$.
– **Public/secret keys:** Select at random $r \in \mathbb{Z}_p^*, X \in \mathbb{G} \setminus \{1\}$. Compute

$$R = g^{-r}, A = e(X, g).$$

The public key is $pk = (R, A)$ and the secret key is $sk = (r, X)$.

– **Sign:** The signature of any string $s \in \{0,1\}^*$ under the public key pk is

$$\sigma = XH(s)^r.$$

– **Verify:** Given a message-signature pair (s, σ), the verification equation is

$$e(\sigma, g)e(H(s), R) = A.$$

If the equation holds, output 1 to represent that purported signature is valid. Else output 0 and reject the purported signature.

– **Encryption:** For a plaintext $m \in \mathbb{G}_T$, randomly select $t \in \mathbb{Z}_p^*$ and compute

$$c_1 = g^t, c_2 = R^t, c_3 = mA^t.$$

– **Decryption:** After receiving a ciphertext (c_1, c_2, c_3), anyone with a valid message-signature pair (s, σ) can extract

$$m = \frac{c_3}{e(\sigma, c_1)e(H(s), c_2)}.$$

The correctness of the proposed ASBB scheme follows from a direct verification. Define $\otimes$ by $(R_1, A_1) \otimes (R_2, A_2) = (R_1R_2, A_1A_2)$ and $\odot$ by $\sigma_1 \odot \sigma_2 = \sigma_1\sigma_2$. For security, we have the following claims in which Claim 2 follows from the definition of $\otimes$ and $\odot$, and the security proof of Claim 3 can be found in the full version of the paper [23].

Theorem 2. *Let $\mathbb{G}$ be a bilinear group of prime order p. For any positive integers n, the following claims hold. (1) The proposed ASBB scheme is (τ', ϵ, n)-aggregatable against non-adaptive chosen message attacks in the random oracle model assuming the decision (τ, ϵ, n)-BDHE assumption holds in $\mathbb{G}$, where $\tau' = \tau + \mathcal{O}((q_H + n^2)\tau_{Exp})$. (2) The proposed ASBB scheme is key-homomorphic. (3) The proposed ASBB scheme is EUF-CMA-Ind-CPA secure in the random oracle model under the Computational Diffie-Hellman (CDH) and Decisional Bilinear Diffie-Hellman (DBDH) assumptions.*

Proof. The aggregatability relies on the decision n-BDHE assumption which is shown to be sound by Boneh et al. [2] in the generic group model. The decision n-BDHE assumption in $\mathbb{G}$ is as follows.

Definition 9. *Let $\mathbb{G}$ be bilinear group of prime order p as defined in Section 4.1 and g, h two independent generators of $\mathbb{G}$. Denote $\overrightarrow{y}_{g,\alpha,n} = (g_1, \cdots, g_n, g_{n+2}, \cdots, g_{2n}) \in \mathbb{G}^{2n-1}$, where $g_i = g^{\alpha^i}$ for some unknown $\alpha \in \mathbb{Z}_p^*$. An algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ε in solving the decision n-BDHE problem if*

$$|\Pr[\mathcal{B}(g, h, \overrightarrow{y}_{g,\alpha,n}, e(g_{n+1}, h)) = 0] - \Pr[\mathcal{B}(g, h, \overrightarrow{y}_{g,\alpha,n}, Z) = 0)]| \geq \varepsilon$$

where the probability is over the random choice of g, h in $\mathbb{G}$, the random choice $\alpha \in \mathbb{Z}_p^*$, the random choice of $Z \in \mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$. We say that the decision (τ, ε, n)-BDHE assumption holds in $\mathbb{G}$ is if no τ-time algorithm has advantage at least ε in solving the decision (τ, ε, n)-BDHE problem.

Based on the decision BDHE assumption, we prove the critical aggregatability of our ASBB scheme. We construct an algorithm $\mathcal{B}$ that uses a decision BDHE challenge to simulate all the requested service for an aggregatability attacker $\mathcal{A}$. Then $\mathcal{B}$ uses the answer bit from $\mathcal{A}$ to solve the the decision BDHE assumption. We illustrate that $\mathcal{B}$ has the same advantage as $\mathcal{A}$ in the same time complexity except for an additive factor, i.e., *the reduction is tight.*

Suppose that $\mathcal{A}$ is a τ-time adversary $\mathcal{A}$ breaking the aggregatability with probability larger than ϵ for a system parameterized with a given n. We build an algorithm $\mathcal{B}$ with advantage ϵ in solving the decision n-BDHE problem in $\mathbb{G}$. $\mathcal{B}$ takes as input a random decision n-BDHE challenge $(g, h, \overrightarrow{y}_{g,\alpha,n}, Z)$, where $\overrightarrow{y}_{g,\alpha,n} = (g_1, \cdots, g_n, g_{n+2}, \cdots, g_{2n})$ and Z is either $e(g_{n+1}, h)$ or a random element of $\mathbb{G}_T$ (recall that $g_i = g^{(\alpha^i)}$). $\mathcal{B}$ proceeds as follows.

Preparation for simulation. For $j = 1, \cdots, n$, $\mathcal{B}$ randomly selects $v_i \in \mathbb{Z}_p$ and computes $h_j = g_j g^{v_j}$. $\mathcal{B}$ randomly selects $i^* \in \{1, \cdots, n\}, a_i, r_i \in \mathbb{Z}_p^*$. Let $S_{i^*} = \{1, \cdots, i^* - 1, i^* + 1, \cdots, n\}$. Compute

$$R_{i^*} = g^{r_{i^*}}(\prod_{k \in S_{i^*}} g_{n+1-k}), \sigma_{i^*,j} = g^{a_{i^*}} g_j^{-r_{i^*}} (\prod\nolimits_{k \in S_{i^*}}^{k \neq j} g_{n+1-k+j}^{-1}) R_{i^*}^{-v_j} (j \neq i^*).$$

For $i \neq i^*$, compute

$$R_i = g^{r_i} g_{n+1-i}^{-1}, \quad \sigma_{i,j} = g^{a_i} g_j^{-r_i} g_{n+1-i+j} R_i^{-v_j} \text{ for } j \neq i.$$

Noting that $g_i^{(\alpha^j)} = g^{(\alpha^{i+j})} = g_{i+j}$ for any i, j, for $j \neq i^*$, we have that

$$\begin{aligned}
&e(\sigma_{i^*,j}, g) e(h_j, R_{i^*}) = e(g^{a_{i^*}} g_j^{-r_{i^*}} (\prod\nolimits_{k \in S_{i^*}}^{k \neq j} g_{n+1-k+j}^{-1}) R_{i^*}^{-v_j}, g) e(g_j g^{v_j}, R_{i^*}) \\
&= e(g^{a_{i^*}} g_j^{-r_{i^*}} (\prod\nolimits_{k \in S_{i^*}}^{k \neq j} g_{n+1-k+j}^{-1}), g) e(g_j, R_{i^*}) \\
&= e(g^{a_{i^*}} g_j^{-r_{i^*}} (\prod\nolimits_{k \in S_{i^*}}^{k \neq j} g_{n+1-k+j}^{-1}), g) e(g_j, g^{r_{i^*}} (\prod\nolimits_{k \in S_{i^*}} g_{n+1-k})) \\
&= e(g^{a_{i^*}} g_j^{-r_{i^*}} (\prod\nolimits_{k \in S_{i^*}}^{k \neq j} g_{n+1-k+j}^{-1}), g) e(g, g_j^{r_{i^*}} (\prod\nolimits_{k \in S_{i^*}} g_{n+1-k+j})) \\
&= e(g^{a_{i^*}}, g) e(g, g_{n+1}) = e(g, g)^{a_{i^*}} e(g, g)^{\alpha^{n+1}} \stackrel{\text{def}}{=} A_{i^*}.
\end{aligned}$$

For $j \neq i (i \neq i^*)$, it holds that

$$\begin{aligned}
&e(\sigma_{i,j}, g) e(h_j, R_i) = e(g^{a_i} g_j^{-r_i} g_{n+1-i+j} R_i^{-v_j}, g) e(g_j g^{v_j}, R_i) \\
&= e(g^{a_i} g_j^{-r_i} g_{n+1-i+j}, g) e(g_j, R_i) = e(g^{a_i} g_j^{-r_i} g_{n+1-i+j}, g) e(g_j, g^{r_i} g_{n+1-i}^{-1}) \\
&= e(g^{a_i}, g) e(g_j^{-r_i} g_{n+1-i+j}, g) e(g, g_j^{r_i} g_{n+1-i+j}^{-1}) = e(g^{a_i}, g) \\
&= e(g, g)^{a_i} \stackrel{\text{def}}{=} A_i.
\end{aligned}$$

Hence, for all $j \neq i (i \in \{1, \cdots, n\})$, we have that

$$e(\sigma_{i,j}, g) e(h_j, R_i) = A_i. \tag{2}$$

After the above preparations, $\mathcal{B}$ runs the aggregatability game with attacker $\mathcal{A}$. During the game, $\mathcal{B}$ will give $\mathcal{A}$ the public system parameters including public keys and $\mathcal{A}$ can ask signatures from $\mathcal{B}$ according to Definition 8. We model the hash function as a random oracle maintained by $\mathcal{B}$ and the $\mathcal{A}$ can ask hash outputs with its chosen queries.

Setup simulation. $\mathcal{B}$ needs to generate n public keys $\{pk_1, \cdots, pk_n\}$. $\mathcal{B}$ sets $pk_i = (R_i, A_i)$ and forwards them to $\mathcal{A}$. Note that r_i, a_i are uniformly distributed in $\mathbb{Z}_p^*$. Hence the simulated public keys have an identical distribution as in the

real world and the simulation is perfect. After the Setup stage, $\mathcal{A}$ has to submit its *distinct* queries $s_j \in \{0,1\}^*(1 \le j \le n)$ in the Education stage to $\mathcal{B}$.

Random oracle simulation. After Setup stage, $\mathcal{A}$ can query the random oracle at any point. Assume that $\mathcal{A}$ queries the random oracle at most q_H times. For each time, $\mathcal{A}$ queries $\mathcal{B}$ with (s'_j, j) for the hash output of s'_j, where $s'_j \in \{0,1\}^*$ and j is a positive integer. Assume that s'_j is fresh (If s'_j has been queried, $\mathcal{B}$ just returns the previous reply for consistence). To simulate the random oracle, $\mathcal{B}$ works as follows. If $1 \le j \le n$ and $s'_j = s_j$, $\mathcal{B}$ sets $H(s'_j) := h_j$; else if $1 \le j \le n$ but $s'_j \ne s_j$ or $n < j \le q_H$, $\mathcal{B}$ randomly selects $h_j \in \mathbb{G}$ and sets $H(s'_j) := h_j$. $\mathcal{B}$ responds with h_j to the query (s'_j, j) from $\mathcal{A}$. Clearly, the simulation of the random oracle is perfect.

Education simulation. For $1 \le j \le n$, $\mathcal{B}$ needs to generate signatures $\sigma_i(s_j)$ such that $\mathtt{verify}(\sigma_i(s_j), s_j, pk_i) = 1 (i \ne j)$. After the above preparation, $\mathcal{B}$ can easily simulate education: When $\mathcal{A}$ requests the signature on s_j under the public key pk_i, $\mathcal{B}$ responds with $\sigma_i(s_j) = \sigma_{i,j}$. From Equation (2), the simulated signatures are well formed and the simulation is also perfect.

Challenge simulation. $\mathcal{B}$ and $\mathcal{A}$ run a standard Ind-CPA game under the the aggregated public encryption key. Denote that $a = a_1 + \cdots + a_n, r = r_1 + \cdots + r_n$. Note that $S_{i^*} = \{1, \cdots, n\} \backslash \{i^*\}$. It follows that the aggregated public encryption key is (R, A) where

$R = R_{i^*} \prod_{k \in S_{i^*}} R_k = g^{r_{i^*}} (\prod_{k \in S_{i^*}} g_{n+1-k}) \prod_{k \in S_{i^*}} g^{r_k} g_{n+1-k}^{-1}$
$= g^{r_{i^*}} \prod_{k \in S_{i^*}} g^{r_k} = g^{\sum_{k=1}^{n} r_k} = g^r,$
$A = A_{i^*} \prod_{k \in S_{i^*}} A_k = e(g,g)^{a_{i^*}} e(g,g)^{\alpha^{n+1}} \prod_{k \in S_{i^*}} e(g,g)^{a_k}$
$= e(g,g)^{\alpha^{n+1}} \prod_{k=1}^{n} e(g,g)^{a_k} = e(g,g)^{\alpha^{n+1}+a}.$

For the aggregated public key (R, A), the decryption keys corresponding to each group member are not revealed. Although neither $\mathcal{B}$ nor $\mathcal{A}$ know the stars $\star$ satisfying $e(\star, g)e(H(s_i), R) = A$, $\mathcal{B}$ knows $r, a \in \mathbb{Z}_p^*$ such that $R = g^r, A = e(g,g)^{\alpha^{n+1}+a}$, where α is unknown. Hence, $\mathcal{B}$ can challenge $\mathcal{A}$ as follows.

When receiving the plaintexts $m_0, m_1 \in \mathbb{G}_T$ chosen by $\mathcal{A}$, $\mathcal{B}$ randomly selects a bit $b \in \{0,1\}$ and computes the challenge ciphertext (c_1^*, c_2^*, c_3^*) where $c_1^* = h, c_2^* = h^r, c_3^* = m_b Z e(g,h)^a$. $\mathcal{B}$ claims that $Z = e(g_{n+1}, h)$ to answer the decision n-BDHE challenge if and only if $\mathcal{A}$'s guess bit $b' = b$.

Success probability: Since g, h are generators, we let $h = g^t$ for some unknown $t \in \mathbb{Z}_p^*$. Hence,

$R^t = (g^r)^t = (g^t)^r = h^r,$
$A^t = (g^r)^t = e(g,g)^{(\alpha^{n+1}+a)t} = e(g, g^t)^{(\alpha^{n+1}+a)}$
$= e(g^{\alpha^{n+1}}, h) e(g,h)^a = e(g_{n+1}, h) e(g,h)^a.$

Therefore, if and only if $Z = e(g_{n+1}, h)$, $(c_1^*, c_2^*, c_3^*) = (g^t, R^t, m_b A^t)$ and (c_1^*, c_2^*, c_3^*) is well formed (*i.e.*, it is a valid ciphertext of m_b under the public key (R, A)). Hence, $\mathcal{B}$ has the same advantage to solve the decision n-BDHE challenge as that of $\mathcal{B}$ breaking the aggregatability of the ASBB scheme.

Time-complexity: In the simulation, $\mathcal{B}$'s overhead is dominated by computing $(\sigma_{i,j}, R_i, A_i)$ for $j \ne i$ and h_j. Computing h_j requires q_H exponentiations in $\mathbb{G}$ (the overhead to sample a random element in $\mathbb{G}$ is about one

exponentiation). Computing $\sigma_{i,j}$ requires $\mathcal{O}(n^2)$ exponentiations in $\mathbb{G}$. Note that $\mathcal{B}$ can compute A_i by an exponentiation in $\mathbb{G}_T$ rather than a pairing computation. Computing R_i, A_i requires $\mathcal{O}(n)$ exponentiations in $\mathbb{G}$ and $\mathbb{G}_T$, respectively. Let $\tau_{\mathtt{Exp}}$ denote the time complexity to compute one exponentiation without differentiation of exponentiations in different groups. Hence, the time complexity of $\mathcal{B}$ is $\tau' = \tau + \mathcal{O}((q_H + n^2)\tau_{\mathtt{Exp}})$. □

4.2 Concrete One-Round ASGKA Protocol

Following the generic construction, it is easy to realize a one-round ASGKA protocol using the instantiated ASBB scheme. Interestingly, we can remove the hash requirements by observing the randomness in the setup stage. That is, we bind the i-th user by randomly choosing $h_i \in \mathbb{G}$ rather than setting $h_i = H(ID_i)$, while other parts are realized literally following from the generic construction.

- Public parameters generation. It is the same as the above ASBB scheme.
- Group setup. Decide the group of the players $\mathbb{P} = \{\mathcal{U}_1, \cdots, \mathcal{U}_n\}$. Randomly choose $h_i \in \mathbb{G}$ for $i = 1, \cdots, n$. h_i can map to $\mathcal{U}_i$ in a natural way, e.g., according to the dictionary order in their binary representation.
- Group key agreement. $\mathcal{U}_i$ randomly chooses $X_i \in \mathbb{G}, r_i \in \mathbb{Z}_p^*$ and publishes

$$\{\sigma_{i,j}, R_i, A_i\}_{i \neq j}, \text{ where } \sigma_{i,j} = X_i h_j^{r_i}, R_i = g^{-r_i}, A_i = e(X_i, g).$$

- Group encryption key derivation. The players share the same group encryption key(R, A):

$$R = \prod_{j=1}^{n} R_j = g^{-\Sigma_{j=1}^{n} r_j}, A = \prod_{j=1}^{n} A_j = e(\prod_{j=1}^{n} X_j, g).$$

- Decryption key derivation. Using the private input (X_i, r_i) during the protocol execution phase, player $\mathcal{U}_i$ can calculate its secret decryption key from the public communication:

$$\sigma_i = X_i h_i^{r_i} \prod_{j=1}^{n, j \neq i} \sigma_{j,i} = \prod_{j=1}^{n} X_j h_i^{r_j} = (\prod_{j=1}^{n} X_j) h_i^{\Sigma_{j=1}^{n} r_j}.$$

- Encryption. For a plaintext $m \in \mathbb{G}_T$, anyone who knows the public parameters and the group encryption key can output the ciphertext $c = (c_1, c_2, c_3)$, where $t \leftarrow \mathbb{Z}_p, c_1 = g^t, c_2 = R^t, c_3 = mA^t$.
- Decryption. Since $e(\sigma_i, g)e(h_i, R) = A$, each player $\mathcal{U}_i$ can decrypt

$$m = \frac{c_3}{e(\sigma_i, c_1)e(h_i, c_2)}.$$

Corollary 1. *The above n-party ASGKA protocol is secure against passive attackers in the standard model under the decision n-BDHE assumption.*

Proof. The proof is a simple combination of Theorems 1 and 2, but since at the group setup stage, $\mathcal{B}$ can simulate $h_j = g_j g^{v_j}$ with a randomly chosen value $v_i \in \mathbb{Z}_p^*$, it does not need the help of a random oracle. The detailed proof is omitted to avoid repetition. □

5 Conclusions and Future Work

We reconsidered the definition of group key agreement and presented the notion of asymmetric group key agreement. Based on a new notion of aggregatable signature-based broadcast, we presented a generic construction of one-round ASGKA protocols, which can also be used as a broadcast scheme but does not need a trusted dealer to distribute secret keys. Finally, efficient ASBB and one-round ASGKA schemes were instantiated using available bilinear pairings. The proposed ASGKA protocol is secure under the decision BDHE assumption without using random oracles. It fills the application gaps left by conventional GKA and broadcast systems. ASGKA being a new notion, it opens numerous avenues for future research such as round-efficient ASGKAs against active attackers, ASGKAs with traitor traceability and (conditional) reductions between ASGKA and conventional GKA protocols.

Acknowledgments and Disclaimer

The authors gratefully acknowledge Professor Colin Boyd for helping to prepare the final paper and anonymous reviewers for their helpful comments. This paper is partly supported by the Spanish Government through projects CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES" and TSI2007-65406-C03-01 "E-AEGIS", by the Australian ARC Discovery Grant DP0877123, and by the Chinese NSF projects 60673071 and 60873268. The fifth author is also partially supported as an ICREA-Acadèmia researcher by the Government of Catalonia. The views of those authors with the UNESCO Chair in Data Privacy do not necessarily reflect the position of UNESCO nor commit that organization.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: A Modular Approach to the Design and Analysis of Authentication and Key Exchange. In: STOC 1998, pp. 419–428. ACM Press, New York (1998)
2. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
3. Boneh, D., Franklin, M.: Identity Based Encryption from the Weil Pairing. SIAM J. of Computing 32(3), 586–615 (2003)
4. Boneh, D., Hamburg, M.: Generalized Identity Based and Broadcast Encryption Schemes. In: Pieprzyk, J. (ed.) Asiacrypt'08. LNCS, vol. 5350, pp. 455–470. Springer, Heidelberg (2008)
5. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
6. Boneh, D., Silverberg, A.: Applications of Multilinear Forms to Cryptography. Contemporary Mathematics 324, 71–90 (2003)

7. Boyd, C., González-Nieto, J.M.: Round-Optimal Contributory Conference Key Agreement. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 161–174. Springer, Heidelberg (2002)
8. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290–309. Springer, Heidelberg (2001)
9. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
10. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably Authenticated Group Diffie-Hellman Key Exchange. In: Samarati, P. (ed.) ACM CCS 2001, pp. 255–264. ACM Press, New York (2001)
11. Burmester, M., Desmedt, Y.G.: A Secure and Efficient Conference Key Distribution System. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
12. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
13. ElGamal, T.: A Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms. IEEE Transaction on Information Theory 31, 467–472 (1985)
14. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
15. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
16. Galbraith, S.D., Rotger, V.: Easy Decision Diffie-Hellman Groups. Journal of Computation and Mathematics 7, 201–218 (2004)
17. Gentry, C.: Certificate-Based Encryption and the Certificate-Revocation Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–291. Springer, Heidelberg (2003)
18. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure against Adaptive Chosen-message Attacks. SIAM J. Computing 17(2), 281–308 (1988)
19. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. J. of Cryptology 17, 263–276 (2004)
20. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
21. Kudla, C., Paterson, K.G.: Modular Security Proofs for Key Agreement Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565. Springer, Heidelberg (2005)
22. Tzeng, W.-G., Tzeng, Z.-J.: Round-Efficient Conference Key Agreement Protocols with Provable Security. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 614–627. Springer, Heidelberg (2000)
23. Wu, Q., Mu, Y., Susilo, W., Qin, B., Domingo-Ferrer, J.: Asymmetric Group Key Agreement. The full version (2009), `http://eprint.iacr.org/`

Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts)

Craig Gentry[1,⋆] and Brent Waters[2,⋆⋆]

[1] Stanford University and IBM
cgentry@cs.stanford.edu
[2] University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. We present new techniques for achieving *adaptive* security in broadcast encryption systems. Previous work on fully collusion resistant broadcast encryption systems with very short ciphertexts was limited to considering only static security.

First, we present a new definition of security that we call *semi-static* security and show a generic "two-key" transformation from semi-statically secure systems to adaptively secure systems that have comparable-size ciphertexts. Using bilinear maps, we then construct broadcast encryption systems that are semi-statically secure in the standard model and have constant-size ciphertexts. Our semi-static constructions work when the number of indices or identifiers in the system is polynomial in the security parameter.

For identity-based broadcast encryption, where the number of potential indices or identifiers may be exponential, we present the first adaptively secure system with sublinear ciphertexts. We prove security in the standard model.

1 Introduction

Broadcast encryption systems [17] allow a sender, who wants to send a message to a dynamically chosen subset $S \subseteq [1, n]$ of users, to construct a ciphertext such that only users in S can decrypt; the sender can then safely transmit this ciphertext over a broadcast channel to all users. It is preferable if the system is *public key* (anybody can encrypt), permits *stateless receivers* (users do not need to update their private keys), and is *fully collusion resistant* (even if all users outside of S collude, they cannot decrypt). Typically in this paper, when we speak of a broadcast encryption system, we will assume that it has these properties. The main challenge in building efficient broadcast systems is to encrypt messages with *short* ciphertexts.

Traditionally, broadcast encryption systems have relied on combinatorial techniques. Such systems include a collusion bound t, where using larger values of

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

⋆ Supported by IBM Fellowship and Herbert Kunzel Stanford Graduate Fellowship.
⋆⋆ Supported by NSF CNS-0716199; and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 171–188, 2009.

t impacts system performance. If an adversary compromises more than t keys, the system would no longer guarantee security even for encryptions solely to uncompromised users. Among systems that are fully collusion resistant, the ciphertext typically grows linearly with either the number of privileged receivers (in the broadcast subset) or the number of revoked users [22,15,20,19,24]. Recently, Boneh, Gentry, and Waters [8] broke through this barrier. They presented new methods for achieving *fully collusion resistant* systems with short (i.e., $\mathcal{O}(\lambda)$, where λ is the security parameter) ciphertexts by applying computational techniques using groups with bilinear maps. However, they used a *static* model of security in which an adversary declares the target set S^* of his challenge ciphertext *before* even seeing the system parameters.

Unfortunately, the weaker static model of security does not capture the power of several types of attackers. Attackers might choose which keys to attempt to compromise and ciphertexts to attack based on the system parameters or the structure of previously compromised keys. To capture general attackers we must use an *adaptive* definition of security.

Adaptive Security. We would like to achieve a system that is provably fully collusion resistant under adaptive attacks. Arguably, this is the "right" model for security in broadcast encryption systems.

Achieving this goal, however, seems challenging. In a security reduction, intuitively, we would expect that a simulation must know all the private keys requested by the attacker, but not know any of the private keys for S^*, the set encrypted to in the challenge ciphertext. Once the public parameters are published, the simulator is essentially bound to what keys it knows. Therefore, in the adaptive setting it might appear that the best we can do in a reduction is to simply guess what keys the adversary might request. Unfortunately, for a system with n users a reduction might guess correctly only a negligible (in n) fraction of the time.

One approach for achieving adaptive security is to apply a hybrid argument. Instead of doing a reduction in one step, one can break the reduction into $n+1$ hybrid experiments $H_0, \ldots, H_n$, such that hybrid games H_i and H_{i+1} are indistinguishable to the adversary. In this reduction in Hybrid H_i the challenge ciphertext is to set $S^* \setminus [1, i]$, where S^* is the challenge specified by the adversary. Since each reduction in the hybrid games lops off only one user at a time, the reduction needs only to guess whether user $i+1$ will be in S^* when distinguishing between H_i and H_{i+1}, thus avoiding an exponential drop-off.

The key leverage that this solution needs is the ability to reduce the target set *anonymously*. This can be done with $\mathcal{O}(\lambda \cdot |S|)$ size ciphertexts. Recently, Boneh and Waters [10] achieved $\mathcal{O}(\lambda \cdot \sqrt{n})$ size ciphertexts. They combine the BGW broadcast techniques with the private linear broadcast techniques of Boneh, Sahai, and Waters [9] (that were originally designed for building traitor tracing techniques). Unfortunately, the $\sqrt{n}$ factor seems to be inherent in this approach with groups that have bilinear (as opposed to say trilinear) maps.

Our Methods. First, we introduce a new general technique for proving systems adaptively secure. The first component of our methodology is the introduction of

the *semi-static* model of security. In the semi-static model of security an attacker must first commit to a set $\tilde{S}$ before setup, but then can later attack any set S^* that is a subset of $\tilde{S}$. This gives the attacker more flexibility than the static model, in which it had to exactly commit to the set it attacks.

At first glance the semi-static model might appear as simply a minor variant of the static model. However, we will also show a generic transformation from semi-static security to adaptive security. Suppose a ciphertext in the semi-static scheme was of size C for a set S of users; then in our transformation the ciphertexts will be of size $2 \cdot C$ plus $|S|$ bits. At the heart of our transformation is a two-key technique where two keys are assigned to each user, but the user is given only one of them. We note that our techniques are partially inspired from those used by Katz and Wang [21] to achieve tight security for IBE systems in the random oracle model.

Using this transformation we might simply hope to prove the BGW system to be semi-statically secure. Unfortunately, the BGW proof of security requires an "exact cancellation" and there is not an obvious way to prove BGW to be semi-statically secure. Instead, we provide two new constructions with constant-size ciphertexts, and prove semi-static security in the standard model. The first construction is a variant of the BGW that still has short ciphertexts, but that requires longer-size private keys. Like the BGW encryption system, we prove our security under the decisional Bilinear Diffie-Hellman Exponent (BDHE) assumption.

Our first construction has two principal limitations. First, it has long private keys. Second, our semi-static transformation works only when $n = \text{poly}(\lambda)$, since the time complexities of the security reductions are at least linear in n. For identity-based broadcast encryption (IBBE), where n may be exponential in λ, we use a different approach.

To solve these problems we use techniques from the Gentry IBE system [18]. We begin by building an "initial" identity-based broadcast encryption system with core component of size $\mathcal{O}(\lambda)$ plus an additional "tag" of size $\mathcal{O}(\lambda \cdot |S|)$. The tag represents a random polynomial in $\mathbb{Z}_p$. The public key is of size $\mathcal{O}(\ell \cdot \lambda)$ for when we can broadcast to at most ℓ users.

While a system with ciphertexts of size $\mathcal{O}(\lambda \cdot |S|)$ is not immediately useful, we can build on this in several ways.

- First, we show that for standard (non-identity-based) broadcast systems we can omit the tag and achieve $\mathcal{O}(\lambda)$ size ciphertexts and private keys while retaining semi-static security.
- Second, we show how in the random oracle model the tag can be generated from a short $O(\lambda)$ size seed and get adaptively secure ID-based broadcast encryption with $O(\lambda)$ size ciphertexts.
- Finally, in the standard model we show how to achieve ID-based encryption with $\mathcal{O}(\lambda \cdot \sqrt{|S|})$ size ciphertexts. In this approach we essentially perform $\sqrt{|S|}$ encryptions to $\sqrt{|S|}$ of the recipients, but share one tag polynomial across all these encryptions.

We prove the security of this base scheme and its derivatives under a new *non-interactive* assumption.

1.1 Related Work

Dodis and Fazio [16] showed how to build an adaptively secure revocation system building upon the techniques of Cramer and Shoup [12] and Naor and Pinkas [23]. In their system the ciphertext size is $\mathcal{O}(\lambda \cdot |R|)$, where R is the set of *revoked* users.

Delerablée, Paillier, and Pointcheval [14] describe a system that is somewhat incomparable to ours and the others discussed here; it allows the adversary to wait until just before each dynamic join operation to declare whether it is joining as an honest or corrupt party (the challenge broadcast is for the honest parties), but then each join operation triggers a change to the public key.

The concept of identity-based broadcast encryption (IBBE) was proposed in [13] (and independently in [27]). This concept is related to identity-based encryption [25], in which the maximal size of a broadcast group is $\ell = 1$. It is also related to multi receiver ID-based KEM (mID-KEM), introduced in [26] and further developed in [4,5,11,2]. We also note that Panjwani [1] considered adaptive corruptions, but in the context of *stateful* protocols such as Logical Key Hierarchy.

2 Adaptive Security in Broadcast Encryption

We present background material on broadcast encryption systems. Then we show our main transformation; we describe how to build adaptive securely broadcast encryption systems from those that are secure against a "semi-static" adversary.

2.1 Broadcast Encryption Systems

We begin by formally defining the notion of security for a public-key broadcast encryption system. For simplicity we define broadcast encryption as a key encapsulation mechanism. In addition, we make our definition general enough to capture identity-based encryption systems.

A broadcast encryption system is made up of four randomized algorithms:

Setup(n, ℓ) Takes as input the number of receivers n and the maximal size $\ell \leq n$ of a broadcast recipient group. It outputs a public/secret key pair $\langle PK, SK \rangle$. (We leave another input, the input security parameter λ, implicit.)

KeyGen(i, SK) Takes as input an index $i \in \{1, \ldots, n\}$ and the secret key SK. It outputs a private key d_i.

Enc(S, PK) Takes as input a subset $S \subseteq \{1, \ldots, n\}$ and a public key PK. If $|S| \leq \ell$, it outputs a pair $\langle \mathrm{Hdr}, K \rangle$ where Hdr is called the header and $K \in \mathcal{K}$ is a message encryption key.

Let $\mathcal{E}_{sym}$ be a symmetric encryption scheme with key-space $\mathcal{K}$, and algorithms *SymEnc* and *SymDec*. Let M be a message to be broadcast to the set S, and let $C_M \stackrel{R}{\leftarrow} \mathit{SymEnc}(K, M)$ be the encryption of M under the symmetric key K. The broadcast to users in S consists of $\langle S, \mathrm{Hdr}, C_M \rangle$.

Dec$(S, i, d_i, \mathrm{Hdr}, PK)$ Takes as input a subset $S \subseteq \{1, \ldots, n\}$, an index $i \in \{1, \ldots, n\}$, a private key d_i for i, a header Hdr, and the public key PK. If $|S| \leq \ell$ and $i \in S$, then the algorithm outputs the message encryption key $K \in \mathcal{K}$. The key K can then be used to decrypt C_M to obtain M.

As usual, we require that the system be correct, namely, that for all $S \subseteq \{1, \ldots, n\}^{\leq \ell}$ and all $i \in S$, if $\langle PK, SK \rangle \xleftarrow{R} \mathit{Setup}(n, \ell)$, $d_i \xleftarrow{R} \mathit{KeyGen}(i, SK)$, and $\langle \mathrm{Hdr}, K \rangle \xleftarrow{R} \mathit{Enc}(S, PK)$, then
$\mathit{Dec}(S, i, d_i, \mathrm{Hdr}, PK) = K$.

Our goal is to illustrate the issues for adaptive security. For simplicity, we define security against chosen plaintext attacks. However, our definitions can readily be extended to reflect chosen-ciphertext attacks.

2.2 Security Definitions

Arguably, the "correct" definition for security in broadcast encryption systems is that of adaptive security. In an adaptively secure system, the adversary is allowed to see PK and then ask for several private keys before choosing the set of indices that it wishes to attack.

Adaptive security in broadcast encryption is defined using the following game between an attack algorithm $\mathcal{A}$ and a challenger. Both the challenger and $\mathcal{A}$ are given n and ℓ as input.

Setup. The challenger runs $\mathit{Setup}(n, \ell)$ to obtain a public key PK, which it gives to the adversary.

Key Query Phase. Algorithm $\mathcal{A}$ adaptively issues private key queries for indices $i \in \{1, \ldots, n\}$.

Challenge. The adversary then specifies a challenge set S^*, such that for all private keys i queried we have that $i \notin S^*$. The challenger sets $\langle \mathrm{Hdr}^*, K_0 \rangle \xleftarrow{R} \mathit{Enc}(S^*, PK)$ and $K_1 \xleftarrow{R} \mathcal{K}$. It sets $b \xleftarrow{R} \{0, 1\}$ and gives (Hdr^*, K_b) to algorithm $\mathcal{A}$.

Guess. Algorithm $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

We define $\mathcal{A}$'s advantage in attacking the broadcast encryption system BE with parameters (n, ℓ) and security parameter λ as

$$\mathsf{AdvBr}_{\mathcal{A}, \mathsf{BE}, n, \ell}(\lambda) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

We may omit the system name when it can be understood from the context.

Definition 1. *We say that a broadcast encryption system* BE *is* adaptively *secure if for all poly-time algorithms* $\mathcal{A}$ *we have that* $\mathsf{AdvBr}_{\mathcal{A}, \mathsf{BE}, n, \ell}(\lambda) = \mathrm{negl}(\lambda)$.

In addition to the adaptive game for broadcast security, we consider two other weaker security notions. The first is *static* security, where the adversary must

commit to the set S^* of identities that it will attack in an "Init" phase before the setup algorithm is run. This is the security definition that is used by recent broadcast encryption systems [8].

We also propose a new security definition called *semi-static* security. In this game the adversary must commit to a set $\tilde{S}$ of indices at the Init phase. The adversary cannot query a private key for any $i \in \tilde{S}$, and it must choose a target group S^* for the challenge ciphertext that is a subset of $\tilde{S}$. A semi-static adversary is weaker than an adaptive adversary, but it is stronger than a static adversary, in that its choice of *which* subset of $\tilde{S}$ to attack can be adaptive.

2.3 Transforming Semi-static Security to Adaptive Security

At first the benefits of achieving semi-static security versus just static security might appear incremental. Indeed, in both games, the adversary is forced to restrict its queries before it even sees the public key.

Despite this apparent shortcoming, we will show that the semi-static security definition is a very useful tool for achieving adaptive security. We will show how to transform any semi-static broadcast encryption scheme to one secure under adaptive attacks with a modest increase in overhead.

Our main idea is to apply a simulation for a two-key technique. In such a system each user will be associated with two potential private keys; however, the authority will give it only one of the two. An encryptor (that does not know which private key the receiver possesses) will need to encrypt the ciphertext twice, once for each key. This technique was used by Katz and Wang [21] to create tightly secure signature and identity-based encryption systems in the random oracle model.

The main benefit is that a simulator will have the private keys for every identity. In the Katz-Wang constructions this enabled tight security reductions. In the context of broadcast encryption, the impact will be much stronger, since trying to guess S^* would otherwise result in an exponential loss of security in the reduction. We now show how to apply the two-key idea to broadcast encryption.

Suppose we are given a semi-static secure broadcast system BE_{SS} with algorithms $Setup_{\mathrm{SS}}$, $KeyGen_{\mathrm{SS}}$, Enc_{SS}, Dec_{SS}. Then we can build our adaptively secure broadcast system BE_A as follows.

Setup(n, ℓ)**:** Run $\langle PK', SK' \rangle \xleftarrow{R} Setup_{\mathrm{SS}}(2n, \ell)$. Set $s \xleftarrow{R} \{0,1\}^n$. Set $PK \leftarrow PK'$ and $SK \leftarrow (SK', s)$. Output $\langle PK, SK \rangle$.

KeyGen(i, SK)**:** Run $d_i' \xleftarrow{R} KeyGen_{\mathrm{SS}}(2i - s_i, SK')$. Set $d_i \leftarrow \langle d_i', s_i \rangle$. Output d_i.

Enc(S, PK)**:** Generate a random set of $|S|$ bits: $t \leftarrow \{t_i \xleftarrow{R} \{0,1\} : i \in S\}$. Generate $K \xleftarrow{R} \mathcal{K}$. Set

$$S_0 \leftarrow \{2i - t_i : i \in S\}\ , \qquad \langle \mathrm{Hdr}_0, \kappa_0 \rangle \xleftarrow{R} Enc_{\mathrm{SS}}(S_0, PK')$$
$$S_1 \leftarrow \{2i - (1 - t_i) : i \in S\}\ , \quad \langle \mathrm{Hdr}_1, \kappa_1 \rangle \xleftarrow{R} Enc_{\mathrm{SS}}(S_1, PK')$$

Set $C_0 \xleftarrow{R} SymEnc(\kappa_0, K), C_1 \xleftarrow{R} SymEnc(\kappa_1, K)$, $\mathrm{Hdr} \leftarrow \langle \mathrm{Hdr}_0, C_0, \mathrm{Hdr}_1, C_1, t \rangle$. Output $\langle \mathrm{Hdr}, K \rangle$.

Dec$(S, i, d_i, \mathrm{Hdr}, PK)$**:** Parse d_i as $\langle d'_i, s_i \rangle$ and Hdr as $\langle \mathrm{Hdr}_0, C_0, \mathrm{Hdr}_1, C_1, t \rangle$. Set S_0 and S_1 as above. Run

$$\kappa_{s_i \oplus t_i} \leftarrow Dec_{\mathrm{SS}}(S_{s_i \oplus t_i}, i, d'_i, \mathrm{Hdr}_{s_i \oplus t_i}, PK')$$

Run $K \leftarrow SymDec(\kappa_{s_i \oplus t_i}, C_{s_i \oplus t_i})$. Output K.

Note that, aside from the string t, the BE_A ciphertext is only about twice as long as a BE_{SS} ciphertext. Suppose that we have a semi-static broadcast encryption system in which ciphertexts are "constant-size" – i.e., $\mathcal{O}(\lambda)$ for security parameter λ. Then, our transformation gives an adaptively secure broadcast encryption system with ciphertexts that are $\mathcal{O}(\lambda + |S|)$, versus $\mathcal{O}(\lambda \cdot |S|)$. In particular, the ciphertext size in BE_A increases by only one bit per additional recipient.

Later, we will describe a semi-static broadcast encryption system in which Hdr contains only two group elements of, say, 200 bits apiece – a total of 400 bits. As an example, suppose we apply the transformation above to this scheme to encrypt to 1000 users, and use AES for the symmetric system. In this case, the Hdr size of the induced adaptively-secure broadcast encryption system is $2 \cdot 400 + 2 \cdot 128 + 1000 = 2056$ bits, versus say $400 \cdot 1000 = 400000$ bits.

It is easy to see that, assuming BE_A is adaptively secure, we can get adaptively secure broadcast encryption system with truly constant-size $\mathcal{O}(\lambda)$ ciphertexts in the random oracle model as follows. Put a hash function $H : \{0,1\}^{\mathcal{O}(\lambda)} \times \{1, \ldots, n\} \rightarrow \{0,1\}$ in the public key. The sender encrypts as before, except that it generates t by setting $u \stackrel{R}{\leftarrow} \{0,1\}^{\mathcal{O}(\lambda)}$ and $t_i \leftarrow H(u, i)$; it replaces t by u in the ciphertext. The recipient decrypts as before, except that it recovers t from u using H.

Alternatively, without random oracles, we get an adaptively secure broadcast encryption system with $\mathcal{O}(\sqrt{\lambda \cdot |S|})$ size ciphertexts from a semi-static system with $\mathcal{O}(\lambda)$ size ciphertexts by partitioning the $|S|$ users into $\sqrt{|S|/\lambda}$ groups of $\sqrt{\lambda \cdot |S|}$ users, and then re-using the same $\sqrt{\lambda \cdot |S|}$-bit string t for every group. Asymptotically, this beats the adaptively-secure system of [10], but often the system above with $\mathcal{O}(\lambda + |S|)$ size ciphertexts will still be preferable in practice. Security follows from the security of the underlying semi-static system by a hybrid argument (omitted).

We now show that BE_A is secure if BE_{SS} is secure.

Theorem 1. *Let $\mathcal{A}$ be an adaptive adversary against BE_A. Then, there exist algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$, each running in about the same time as $\mathcal{A}$, such that*

$$\begin{aligned}\mathsf{AdvBr}_{\mathcal{A},\mathsf{BE}_A,n,\ell}(\lambda) \leq\ & \mathsf{AdvBrSS}_{\mathcal{B}_1,\mathsf{BE}_{SS},2n,\ell}(\lambda) \ +\ \mathsf{AdvBrSS}_{\mathcal{B}_2,\mathsf{BE}_{SS},2n,\ell}(\lambda) \\ & +\ \mathsf{AdvSym}_{\mathcal{B}_3,\mathcal{E}_{sym}}(\lambda) \ +\ \mathsf{AdvSym}_{\mathcal{B}_4,\mathcal{E}_{sym}}(\lambda)\end{aligned}$$

Proof. We present the proof as a sequence of games. Let W_i denote the event that $\mathcal{A}$ wins game i.

Game 0. The first game is identical to the adaptive security game given above. Thus,

$$\left|\Pr[W_0] - \frac{1}{2}\right| = \mathsf{AdvBr}_{\mathcal{A},\mathsf{BE}_A,n,\ell}(\lambda) \tag{1}$$

Game 1. Game 1 is identical to Game 0, except that the challenger generates C_0 in the challenge ciphertext as follows: set $\kappa_0^\dagger \xleftarrow{R} \mathcal{K}$ and then $C_0 \xleftarrow{R} \mathit{SymEnc}(\kappa_0^\dagger, K_0)$.

We claim that there exists an algorithm $\mathcal{B}_1$, whose running time is about the same as $\mathcal{A}$, such that

$$|\Pr[W_1] - \Pr[W_0]| = \mathsf{AdvBrSS}_{\mathcal{B}_1,\mathsf{BE}_{SS},2n,\ell}(\lambda) \tag{2}$$

To break BE_{SS}, $\mathcal{B}_1$ sets $s \xleftarrow{R} \{0,1\}^n$ and $\tilde{S} \leftarrow \{2i - (1 - s_i) : i \in \{1,\ldots,n\}\}$. It sends $\tilde{S}$ to the challenger, which sends back PK'. $\mathcal{B}$ sets $PK \leftarrow PK'$ and forwards PK to $\mathcal{A}$.

When $\mathcal{A}$ queries the BE_A private key for $i \in \{1,\ldots,n\}$, $\mathcal{B}$ queries the challenger for the BE_{SS} private key for $2i - s_i$. The challenger sends back d_i'; $\mathcal{B}$ sends (d_i', s_i) to $\mathcal{A}$.

$\mathcal{A}$ requests a challenge ciphertext on some $S^* \subseteq \{1,\ldots,n\}$. $\mathcal{B}$ sets $t \leftarrow \{t_i \leftarrow 1 - s_i : i \in S^*\}$. It sets $S_0 \leftarrow \{2i - t_i : i \in S^*\}$ and $S_1 \leftarrow \{2i - (1 - t_i) : i \in S^*\}$, and queries the challenger for a challenge ciphertext on S_0. The challenger sends back $(\mathrm{Hdr}_0, \kappa_0^{(b)})$, where b denotes the bit flipped by the challenger. $\mathcal{B}$ sets $(\mathrm{Hdr}_1, \kappa_1) \xleftarrow{R} \mathit{Enc}(S_1, PK')$. It generates $K_0, K_1 \xleftarrow{R} \mathcal{K}$, $b^\dagger \xleftarrow{R} \{0,1\}$, $C_0 \xleftarrow{R} \mathit{SymEnc}(\kappa_0^{(b)}, K_0)$ and $C_1 \xleftarrow{R} \mathit{SymEnc}(\kappa_1, K_0)$. It sets $\mathrm{Hdr} \leftarrow \langle \mathrm{Hdr}_0, C_0, \mathrm{Hdr}_1, C_1, t\rangle$. It sends $(\mathrm{Hdr}, K_{b^\dagger})$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs a bit b'. If $b' = b^\dagger$, $\mathcal{B}$ sends 0 to the challenger; else, it sends 1.

If $b = 0$, $\mathcal{A}$'s view is as in Game 0. The private keys sent by $\mathcal{B}$ are appropriately distributed. The string t appears to be uniformly random, since $\mathcal{A}$'s private key queries reveal only the values of s_i for $i \notin S^*$. Also, $\kappa_0^{(0)}$ is generated correctly, and so the dependent values are as well. If $b = 1$, $\mathcal{A}$'s view is as in Game 1. The claim follows.

Game 2. Game 2 is identical to Game 1, except that the challenger sets $\kappa_1 \xleftarrow{R} \mathcal{K}$ when constructing the challenge ciphertext. By an analysis similar to above, we conclude that there exists an algorithm $\mathcal{B}_2$, which runs in about the same time as $\mathcal{A}$, for which

$$|\Pr[W_2] - \Pr[W_1]| = \mathsf{AdvBrSS}_{\mathcal{B}_2,\mathsf{BE}_{SS},2n,\ell}(\lambda) \tag{3}$$

Game 3. Game 3 is identical to Game 2, except that the challenger sets $K_0^\dagger \xleftarrow{R} \mathcal{K}$ and $C_0 \xleftarrow{R} \mathit{SymEnc}(\kappa_0^\dagger, K_0^\dagger)$. We claim that there exists an algorithm $\mathcal{B}_3$, which runs in about the same time as $\mathcal{A}$, for which

$$|\Pr[W_3] - \Pr[W_2]| = \mathsf{AdvSym}_{\mathcal{B}_3,\mathcal{E}_{sym}}(\lambda) \tag{4}$$

This follows, since it is straightforward to construct $\mathcal{B}_3$ as an algorithm that attacks the semantic security of $\mathcal{E}_{sym}$.

Game 4. Game 4 is identical to Game 3, except that the challenger sets $K_1^\dagger \stackrel{R}{\leftarrow} \mathcal{K}$ and $C_1 \stackrel{R}{\leftarrow} \mathit{SymEnc}(\kappa_1^\dagger, K_1^\dagger)$. As above, we obtain

$$|\Pr[W_4] - \Pr[W_3]| = \mathsf{AdvSym}_{\mathcal{B}_4, \mathcal{E}_{sym}}(\lambda) \tag{5}$$

Finally, the theorem follows if the following claim is true:

$$\left| \Pr[W_4] - \frac{1}{2} \right| = 0 \tag{6}$$

This claim follows since, in Game 4, Hdr is independent of K_b, and hence b.

3 BE Construction with Small Ciphertexts

Now that we have our transformation of semi-static security to adaptive security, we would like to leverage it to create new adaptively secure broadcast encryption systems. One obvious candidate to examine is the Boneh-Gentry-Waters [8] broadcast encryption system. Unfortunately, it was proven only to be statically secure and there does not appear to be an obvious way to make the proof semi-static.[1]

To prove semi-static security we will need to use a variant of the BGW system. We first describe our construction. Then we describe the decisional-BDHE assumption (the same one used by BGW). Then we prove our system to be semi-statically secure under this assumption.

3.1 Our Construction

Let $\mathit{GroupGen}(\lambda, n)$ be an algorithm that, on input security parameter λ, generates groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order $p = p(\lambda, n) > n$ with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Setup(n, n)**:** Run $\langle \mathbb{G}, \mathbb{G}_T, e \rangle \stackrel{R}{\leftarrow} \mathit{GroupGen}(\lambda, n)$. Set $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $g, h_1, \ldots, h_n \stackrel{R}{\leftarrow} \mathbb{G}^{n+1}$. Set PK to include a description of $\langle \mathbb{G}, \mathbb{G}_T, e \rangle$, as well as

$$g \ , \quad e(g,g)^\alpha \ , \quad h_1 \ , \ \ldots , \quad h_n .$$

The secret key is $SK \leftarrow g^\alpha$. Output $\langle PK, SK \rangle$

KeyGen(i, SK)**:** Set $r_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and output

$$d_i \leftarrow \langle d_{i,0}, \ldots, d_{i,n} \rangle \ \text{ where } \ d_{i,0} \leftarrow g^{-r_i} \ , \quad d_{i,i} \leftarrow g^\alpha h_i^{r_i} \ , \quad \forall_{j \neq i} \ d_{i,j} \leftarrow h_j^{r_i}$$

[1] The BGW reduction depends upon an *exact* cancellation between a value embedded by the simulator in the parameters and a function of the target set S^*.

Enc(S, PK)**:** Set $t \xleftarrow{R} \mathbb{Z}_p$ and

$$\text{Hdr} \leftarrow \langle C_1, C_2 \rangle \quad \text{where} \quad C_1 \leftarrow g^t \ , \quad C_2 \leftarrow (\prod_{j \in S} h_j)^t$$

Set $K \leftarrow e(g,g)^{\alpha \cdot t}$. Output $\langle \text{Hdr}, K \rangle$.

Dec$(S, i, d_i, \text{Hdr}, PK)$**:** If $i \in S$, parse d_i as $\langle d_{i,0}, \ldots, d_{i,n} \rangle$ and Hdr as $\langle C_1, C_2 \rangle$ and output

$$K \leftarrow e(d_{i,i} \cdot \prod_{j \in S \setminus \{i\}} d_{i,j}, C_1) \cdot e(d_{i,0}, C_2)$$

Correctness: We check that decryption recovers the correct value of K.

$$\begin{aligned} e(d_{i,i} \cdot \prod_{j \in S \setminus \{i\}} d_{i,j}, C_1) \cdot e(d_{i,0}, C_2) = \ & e(g^\alpha \cdot (\prod_{j \in S} h_j)^{r_i}, g^t) \cdot e(g^{-r_i}, (\prod_{j \in S} h_j)^t) \\ & = e(g,g)^{\alpha \cdot t} \end{aligned}$$

as required.

3.2 The BDHE Assumption

We base the security of the above system on the decision BDHE assumption, used in [8]. The decision BDHE problem is as follows.

Definition 2 (Decision BDHE problem (for m)). *Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of order p with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and let g be a generator for $\mathbb{G}$. Set $a, s \xleftarrow{R} \mathbb{Z}_p^*$ and $b \xleftarrow{R} \{0,1\}$. If $b = 0$, set $Z \leftarrow e(g,g)^{a^{m+1} \cdot s}$; else, set $Z \xleftarrow{R} \mathbb{G}_T$. The problem instance consists of g^s, Z, and the set*

$$\{g^{\alpha^i} : i \in [0, m] \cup [m+2, 2m]\}$$

The problem is to guess b.

We define $\mathsf{AdvBDHE}_{\mathcal{A},m}(\lambda)$ in the expected way. We have the following theorem.

Theorem 2. *Let $\mathcal{A}$ be a semi-static adversary against the above system. Then, there is an algorithm $\mathcal{B}$, which runs in about the same time as $\mathcal{A}$, such that*

$$\mathsf{AdvBrSS}_{\mathcal{A},n,n}(\lambda) = \mathsf{AdvBDHE}_{\mathcal{B},n}(\lambda)$$

We provide the proof in Appendix A.

3.3 Semi-static BE with Small Ciphertexts and Private Keys

In the semi-static system described in Section 3, the public key and private keys are of size $\mathcal{O}(\lambda \cdot n)$. However, we have an alternative construction that has a public key of size $\mathcal{O}(\lambda \cdot \ell)$ and constant-sized private keys (i.e., $\mathcal{O}(\lambda)$). This construction is a special case of the identity-based broadcast encryption system that we provide in Section 4.1. We provide more details in Section 4.3.

4 Identity-Based BE with Small Ciphertexts and Private Keys

The essential property of an *identity-based* broadcast encryption (IBBE) system is that it remains efficient when n is exponential in the security parameter λ. Adaptive security is even more challenging in this setting. In particular, our semi-static constructions do not give adaptively secure IBBE, since the time complexities of the reduction algorithms are at least linear in n.

Here we first describe an initial IBBE system with adaptive security, where the ciphertext size is constant aside from a random "tag" that has length $\mathcal{O}(\lambda \cdot |S|)$. This long tag is needed by the simulator to handle the fact that the adversary chooses the target set S^* adaptively. The public key has size $\mathcal{O}(\lambda \cdot \ell)$, and private keys are constant size (i.e., $\mathcal{O}(\lambda)$). This system is an extension of Gentry's IBE system [18].

At first, a system with such a long tag appears to be pointless. However, there are several ways to address this apparent problem. First, for polynomial-size n, we show that the system is semi-statically secure if we replace the random tag with a *constant* tag; the ciphertext size then becomes constant. Second, we make the straightforward observation that, in the random oracle model, we obtain an adaptively secure IBBE system with constant-size ciphertexts if we generate the tag from the random oracle. Finally, we construct an adaptively secure IBBE system (in the standard model) that, for a recipient group of size $k \leq \ell$, has $\mathcal{O}(\lambda \cdot \sqrt{k})$-size ciphertexts, a $\mathcal{O}(\lambda \cdot \sqrt{\ell})$-size public key, and still constant-size private keys, by reusing the same $\mathcal{O}(\lambda \cdot \sqrt{k})$-size tag in $\mathcal{O}(\sqrt{k})$ separate sub ciphertexts from the initial system. As far as we know, this is the first IBBE system with sub-linear ciphertexts secure against adaptive adversaries.

4.1 An Initial IBBE Construction

Let $\mathit{GroupGen}(\lambda, n, \ell)$ be an algorithm that outputs suitable bilinear group parameters $\langle \mathbb{G}, \mathbb{G}_T, e \rangle$, where $\mathbb{G}$ is of order $p \geq n + \ell$.

Setup(n, ℓ)**:** Run $\langle \mathbb{G}, \mathbb{G}_T, e \rangle \xleftarrow{R} \mathit{GroupGen}(\lambda, n, \ell)$. Set $g_1, g_2 \xleftarrow{R} \mathbb{G}$. Set $\alpha, \beta, \gamma \xleftarrow{R} \mathbb{Z}_p$. Set $\hat{g}_1 \leftarrow g_1^{\beta}$ and $\hat{g}_2 \leftarrow g_2^{\beta}$. PK contains a description of $\langle \mathbb{G}, \mathbb{G}_T, e \rangle$, the parameters n and ℓ, along with g_1^{γ}, $g_1^{\gamma \cdot \alpha}$ and the set

$$\{g_1^{\alpha^j}, \hat{g}_1^{\alpha^j}, g_2^{\alpha^k}, \hat{g}_2^{\alpha^k} : j \in [0, \ell], k \in [0, \ell - 2]\}$$

Generate a random key κ for a PRF $\Psi : [1, n] \to \mathbb{Z}_p$. The private key is $SK \leftarrow (\alpha, \gamma, \kappa)$.

KeyGen(i, SK)**:** Similar to Gentry's IBE system, set $r_i \leftarrow \Psi_\kappa(i)$ and output the private key

$$d_i \leftarrow \langle r_i, h_i \rangle \ , \quad \text{where} \quad h_i \leftarrow g_2^{\frac{\gamma - r_i}{\alpha - i}}$$

Enc(S, PK)**:** Run $\tau \xleftarrow{R} \mathit{TagGen}(S, PK)$. Output $\langle \mathrm{Hdr}, K \rangle \xleftarrow{R} \mathit{TagEncrypt}(\tau, S, PK)$.

TagGen(S, PK)**:** Let $k = |S|$. Set $F(x) \in \mathbb{Z}_p[x]$ to be a random $(\ell - 1)$-degree polynomial such that $F(n + j) = 1$ for $j \in [k + 1, \ell]$. Output $\tau \leftarrow F(x)$. Note that τ can be expressed by k values in $\mathbb{Z}_p$ – e.g., $\{F(i) : i \in S\}$; $F(x)$ can be interpolated from these values and $\{F(n + j) = 1 : j \in [k + 1, \ell]\}$.

TagEncrypt(τ, S, PK)**:** Parse τ as $F(x)$ and S as $\{i_1, \ldots, i_k\}$. Set $i_j \leftarrow n + j$ for $j \in [k + 1, \ell]$. Set $P(x) = \prod_{j=1}^{\ell}(x - i_j)$. Set $t \xleftarrow{R} \mathbb{Z}_p$ and set $K \leftarrow e(g_1, \hat{g}_2)^{\gamma \cdot \alpha^{\ell-1} \cdot t}$. Next, set

$$\text{Hdr} \leftarrow \langle C_1, \ldots, C_4 \rangle \leftarrow \langle \hat{g}_1^{P(\alpha) \cdot t}, \quad g_1^{\gamma \cdot t}, \quad g_1^{F(\alpha) \cdot t}, \quad e(g_1, \hat{g}_2)^{\alpha^{\ell-1} \cdot F(\alpha) \cdot t} \rangle .$$

Output $\langle \tau, \text{Hdr}, K \rangle$.

Dec$(S, i, d_i, \tau, \text{Hdr}, PK)$**:** Suppose $i \in S = \{i_1, \ldots, i_k\}$. Parse d_i as $\langle r_i, h_i \rangle$, τ as $F(x)$, and Hdr as $\langle C_1, \ldots, C_4 \rangle$. Define $P(x)$ as above. Let

$$P_i(x) = x^{\ell-1} - \frac{P(x)}{(x - i)}, \quad F_i(x) = \frac{F(x) - F(i)}{x - i}, \quad \text{and} \quad e_i = -\frac{r_i}{F(i)} .$$

Set

$$K \leftarrow e(C_1, h_i \cdot g_2^{e_i \cdot F_i(\alpha)}) \cdot e(C_2 \cdot C_3^{e_i}, \hat{g}_2^{P_i(\alpha)}) / C_4^{e_i} \tag{7}$$

Note that the recipient can compute $g_2^{F_i(\alpha)}$ and $\hat{g}_2^{P_i(\alpha)}$ from PK, since $F_i(x)$ and $P_i(x)$ are polynomials of degree $\ell - 2$.

Correctness: We verify that decryption recovers the message. First, we note that $K = K_1 \cdot K_2$, where we gather the terms containing a γ in K_1, and the other terms in K_2. (Recall $h_i = g_2^{\gamma/(\alpha - i)} \cdot g_2^{-r_i/(\alpha - i)}$.)

$$K_1 = e(C_1, g_2^{\gamma})^{1/(\alpha - i)} \cdot e(C_2, \hat{g}_2^{P_i(\alpha)})$$
$$K_2 = e(C_1, g_2^{-r_i/(\alpha - i) + e_i \cdot F_i(\alpha)}) \cdot e(C_3, \hat{g}_2^{P_i(\alpha)})^{e_i} / C_4^{e_i}$$

We have that

$$K_1^{1/t} = e(g_1, \hat{g}_2)^{\gamma(P(\alpha)/(\alpha - i) + P_i(\alpha))} = e(g_1, \hat{g}_2)^{\gamma \cdot \alpha^{\ell-1}}$$

We also have that

$$\begin{aligned} K_2^{1/t} &= e(g_1, \hat{g}_2)^{-r_i \cdot P(\alpha)/(\alpha - i) + e_i \cdot P(\alpha) \cdot F_i(\alpha) + e_i \cdot P_i(\alpha) \cdot F(\alpha) - e_i \cdot \alpha^{\ell-1} \cdot F(\alpha)} \\ &= e(g_1, \hat{g}_2)^{e_i \cdot P(\alpha) \cdot F(\alpha)/(\alpha - i) + e_i \cdot P_i(\alpha) \cdot F(\alpha) - e_i \cdot \alpha^{\ell-1} \cdot F(\alpha)} \\ &= e(g_1, \hat{g}_2)^{e_i \cdot F(\alpha)\left(P(\alpha)/(\alpha - i) + P_i(\alpha) - \alpha^{\ell-1}\right)} \\ &= e(g_1, \hat{g}_2)^0 \quad = \quad 1 \end{aligned}$$

as required.

4.2 Security of the Initial IBBE Construction

Below, we define a class of assumptions that is narrower than the general bilinear DH exponent "uber-assumption" defined by Boneh et al. [6], but broad enough to cover some frequently used assumptions. One reason that we think carving out this class of assumptions is useful is that it is much easier to glance at an assumption in this class and verify that it at least superficially makes sense than it is for some of the wilder assumptions within general BDHE.

Definition 3 (The Decision BDHE Sum Problem for (S, m)). *Fix $S \subset \mathbb{Z}$ and $m \in \mathbb{Z} \setminus (S + S)$. Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of order p with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and let g be a generator for $\mathbb{G}$. Set $\alpha \xleftarrow{R} \mathbb{Z}_p^*$ and $b \xleftarrow{R} \{0, 1\}$. If $b = 0$, set $Z \leftarrow e(g, g)^{\alpha^m}$; otherwise, set $Z \xleftarrow{R} \mathbb{G}_T$. Output*

$$\{g^{\alpha^i} : i \in S\} \quad \text{and} \quad Z$$

The problem is to guess b.

In the decision n-BDHI problem, $S = [0, n]$ and $m = -1$. One can reduce the Decision BDHE Sum problem for $S = [0, n] \cup [n + 2, 2n] \cup [3n]$ and $m = 4n + 1$ to the decision BDHE problem for n – i.e., s in the BDHE problem is replaced by α^{3n}.

Although we do not use it in this paper, we mention an obvious (possibly easier) variant of the problem:

Definition 4 (The Decision BDHE Sum Problem for (S, m) (variant)). *As above, except Z is replaced in the instance by random $(z_1, z_2) \in \mathbb{G}^2$ satisfying $e(z_1, z_2) = Z$.*

A recent paper [3] builds the first adaptively secure hierarchical identity based encryption (HIBE) system that allows a polynomial number of levels by building on our IBBE system and using this variant of the Decision BDHE Sum problem.

We base the security of our system on the Decision BDHE Sum problem for $m = 4d + 4\ell - 1$ and

$$\begin{aligned} S = [0, \ell - 2] \cup [d + \ell, 2d + \ell - 1] \cup [2d + 2\ell, 2d + 3\ell - 1] \\ \cup [3d + 3\ell, 4d + 3\ell] \cup [4d + 4\ell, 5d + 4\ell + 1] \end{aligned}$$

where $d = q + 2\ell$, q and ℓ non-negative. We define $\mathsf{AdvBDHES}_{\mathcal{A},q,\ell}(\lambda)$ in the expected way, using these particular values of S and m.

We have the following theorem.

Theorem 3. *Let $\mathcal{A}$ be an adaptive adversary against the above initial IBBE system that makes at most q queries. Then, there exist algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathsf{AdvBr}_{\mathcal{A},n,\ell}(\lambda) \leq \mathsf{AdvPRF}_{\mathcal{B}_1,\Psi}(\lambda) \ + \ \mathsf{AdvBDHES}_{\mathcal{B}_2,q,\ell}(\lambda) \ + \ (\ell + 2)/p \quad (8)$$

where $\mathcal{B}_1$ runs in about the same time as $\mathcal{A}$, and $\mathcal{B}_2$ runs in time $t(\mathcal{A}) + \mathcal{O}((q + \ell)^2 \cdot \lambda^3)$, assuming exponentiations take time $\mathcal{O}(\lambda^3)$.

We provide the proof in Appendix B.

4.3 Variants of the IBBE Construction

Semi-Static BE with Constant-Size Ciphertexts and Private Keys. When $n = \text{poly}(\lambda)$, we obtain a semi-statically secure variant of the above system with constant-size ciphertexts by making the following simple change.

TagGen(S, PK)**:** Output $\tau \leftarrow F(x) \leftarrow 1$.

Since τ is always 1, we do not need to include it in the ciphertext. Also, some terms in PK become unnecessary – in particular, $\{g_2^{\alpha^i} : i \in [0, \ell - 2]\}$.

We have the following theorem. Let $q = n$.

Theorem 4. *Let $\mathcal{A}$ be a semi-static adversary against the above system. Then, there exist algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathsf{AdvBr}_{\mathcal{A},n,\ell}(\lambda) \leq \mathsf{AdvPRF}_{\mathcal{B}_1,\Psi}(\lambda) \ + \ \mathsf{AdvBDHES}_{\mathcal{B}_2,q,\ell}(\lambda) \ + \ (\ell + 2)/p \quad (9)$$

where $\mathcal{B}_1$ runs in about the same time as $\mathcal{A}$ and $\mathcal{B}_2$ runs in time $t(\mathcal{A}) + \mathcal{O}((q + \ell)^2 \cdot \lambda^3)$, assuming exponentiations take time $\mathcal{O}(\lambda^3)$.

We prove this simultaneously with Theorem 3 in Appendix B.

Adaptively Secure IBBE with Constant-Size Ciphertexts in the ROM. In the random oracle model, the obvious way to modify the initial IBBE system to obtain constant-size ciphertexts is to generate τ using a hash function $H : \{0,1\}^{\mathcal{O}(\lambda)} \times [1, n] \rightarrow \mathbb{Z}_p$. In particular, we make the following modification.

TagGen(S, PK)**:** Output $\tau \leftarrow \{0,1\}^{\mathcal{O}(\lambda)}$.

In *TagEncrypt* and *Dec*, $F(x)$ is set to be the $(\ell - 1)$-degree polynomial that interpolates $F(i) = H(\tau, i)$ for $i \in S$ and $F(i) = 1$ for $i \in [n+j]$ with $j \in [k+1, \ell]$. The ciphertext size is constant, since the size of τ is constant (i.e., $\mathcal{O}(\lambda)$). We omit the easy tight reduction from an adversary that breaks the initial system to an adversary that breaks this system.

Adaptively Secure IBBE with Sublinear-Size Ciphertexts. Let $\ell = \ell_1 \cdot \ell_2$. Below, we describe a system that builds on the initial IBBE system and allows one to encrypt to a set S with $|S| = k_1 \cdot k_2$, $k_1 \leq \ell_1$, $k_2 \leq \ell_2$.

Setup$_{\mathbf{SL}}(n, \ell)$**:** Run $(PK', SK') \leftarrow \mathit{Setup}(n, \ell_2)$. Set $PK \leftarrow (PK', \ell_1)$ and $SK \leftarrow SK'$. Output $\langle PK, SK \rangle$.

KeyGen$_{\mathbf{SL}}(i, SK)$**:** Run $d_i \xleftarrow{R} \mathit{KeyGen}(i, SK')$. Output d_i.

Encrypt$_{\mathbf{SL}}(S, PK)$**:** Partition S into $k_1 \leq \ell_1$ sets $\langle S_1, \ldots, S_{k_1} \rangle$ of size $k_2 \leq \ell_2$. Run $\tau \xleftarrow{R} \mathit{TagGen}(S_1, PK')$. Generate $K \xleftarrow{R} \mathcal{K}$. For $j \in [1, k_1]$, set

$$\langle \text{Hdr}_j, \kappa_j \rangle \xleftarrow{R} \mathit{TagEncrypt}(\tau, S_j, PK') \ , \quad c_j \leftarrow \mathit{SymEnc}(\kappa_j, K)$$

Set $\text{Hdr} \leftarrow \langle \text{Hdr}_1, c_1, \ldots, \text{Hdr}_{k_1}, c_{k_1} \rangle$. Output $\langle \tau, \text{Hdr}, K \rangle$.

***Decrypt*$_{SL}$**$(S, i, d_i, \tau, \mathrm{Hdr}, PK)$**:** Parse Hdr as $\langle \mathrm{Hdr}_1, c_1, \ldots, \mathrm{Hdr}_{k_1}, c_{k_1} \rangle$ and S as $\langle S_1, \ldots, S_{k_1} \rangle$. Suppose $i \in S_j$. Run

$$\kappa_j \leftarrow Dec(S_j, i, d_i, \tau, \mathrm{Hdr}_j, PK') \quad \text{and} \quad K \leftarrow SymDec(\kappa_j, c_j)$$

Output K.

We have the following theorem.

Theorem 5. *Let $\mathcal{A}$ be an adaptive adversary against this system that makes at most q queries. Then, there exist algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$, the former being an adversary against the initial IBBE system that makes at most q queries, each algorithm running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{AdvBr}_{\mathcal{A},n,\ell}(\lambda) \leq \ell_1 \cdot \Big(\mathsf{AdvBr}_{\mathcal{B}_1,n,\ell_2}(\lambda) \ + \ \mathsf{AdvSym}_{\mathcal{B}_2,\mathcal{E}_{sym}}(\lambda) \Big) \tag{10}$$

As before $\mathcal{E}_{sym}$ is a symmetric encryption scheme. We omit the proof, since it is a simple hybrid argument similar to the proof of Theorem 1.

It is easy to handle the case where $|S|$ cannot be expressed as a product $k_1 \cdot k_2$ with $k_1, k_2 = \mathcal{O}(\sqrt{|S|})$. Let S' consist of the first $k_1 \cdot k_2$ identities in S, where $k_1 = k_2 = \lfloor \sqrt{|S|} \rfloor$. Encrypt to S' using the above system, and to $S \setminus S'$ using any reasonable system – e.g., the initial system. The overall size of the ciphertext is still $\mathcal{O}(\lambda \cdot \sqrt{|S|})$. One can prove the security of this double encryption by a sequence of games similar to the proof of Theorem 1.

References

1. Panjwani, S.: Tackling adaptive corruptions in multicast encryption protocols. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 21–40. Springer, Heidelberg (2007)
2. Abdalla, M., Kiltz, E., Neven, G.: Generalized Key Delegation for Hierarchical Identity-Based Encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 139–154. Springer, Heidelberg (2007)
3. Anonymous. Hierarchical Identity Based Encryption with Polynomially Many Levels (Manuscript, 2008)
4. Baek, J., Safavi-Naini, R., Susilo, W.: Efficient Multi-receiver Identity-Based Encryption and Its Application to Broadcast Encryption. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 380–397. Springer, Heidelberg (2005)
5. Barbosa, M., Farshim, P.: Efficient Identity-Based Key Encapsulation to Multiple Parties. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 428–441. Springer, Heidelberg (2005)
6. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
7. Boneh, D., Gentry, C., Hamburg, M.: Space Efficient Identify Based Encryption without Pairings. In: FOCS 2007 (2007)
8. Boneh, D., Gentry, C., Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)

9. Boneh, D., Sahai, A., Waters, B.: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
10. Boneh, D., Waters, B.: A Fully Collusion Resistant Broadcast, Trace, and Revoke System. In: CCS 2006 (2006)
11. Chatterjee, S., Sarkar, P.: Multi-receiver Identity-Based Key Encapsulation with Shortened Ciphertext. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 394–408. Springer, Heidelberg (2006)
12. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
13. Delerablée, C.: Identity-Based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007)
14. Delerablée, C., Paillier, P., Pointcheval, D.: Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 39–59. Springer, Heidelberg (2007)
15. Dodis, Y., Fazio, N.: Public Key Broadcast Encryption for Stateless Receivers. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
16. Dodis, Y., Fazio, N.: Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 100–115. Springer, Heidelberg (2002)
17. Fiat, A., Naor, M.: Broadcast Encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
18. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
19. Goodrich, M.T., Sun, J.Z., Tamassia, R.: Efficient Tree-Based Revocation in Groups of Low-State Devices. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 511–527. Springer, Heidelberg (2004)
20. Halevy, D., Shamir, A.: The LSD Broadcast Encryption Scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
21. Katz, J., Wang, N.: Efficiency Improvements for Signature Schemes with Tight Security Reductions. In: CCS 2003 (2003)
22. Naor, D., Naor, M., Lotspiech, J.: Revocation and Tracing Schemes for Stateless Receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
23. Naor, M., Pinkas, B.: Efficient Trace and Revoke Schemes. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 1–20. Springer, Heidelberg (2001)
24. Sharmila Deva Selvi, S., Sree Vivek, S., Gopalakrishnan, R., Karuturi, N.N., Pandu Rangan, C.: Provably Secure ID-Based Broadcast Signcryption (IBBSC) Scheme. Eprint 2008/225
25. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
26. Smart, N.P.: Efficient Key Encapsulation to Multiple Parties. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 208–219. Springer, Heidelberg (2005)
27. Sakai, R., Furukawa, J.: Identity-Based Broadcast Encryption. Eprint 2007/217

A Proof of Theorem 2

$\mathcal{B}$ receives the problem instance, which includes g^s, Z, and the set

$$\{g^{a^i} : i \in [0,n] \cup [n+2, 2n]\}$$

Init $\mathcal{A}$ commits to a set $\tilde{S} \subseteq [1,n]$.

Setup $\mathcal{B}$ generates $y_0, \ldots, y_n \stackrel{R}{\leftarrow} \mathbb{Z}_p$. It sets

$$\begin{array}{rcll} h_i & \leftarrow & g^{y_i} & \text{for } i \in \tilde{S} \\ h_i & \leftarrow & g^{y_i + a^i} & \text{for } i \in [1,n] \setminus \tilde{S} \end{array}$$

Formally, $\mathcal{B}$ sets $\alpha \leftarrow y_0 \cdot a^{n+1}$. It sets PK to include a description of $\langle \mathbb{G}, \mathbb{G}_T, e \rangle$, as well as

$$g \ , \quad e(g,g)^{\alpha} \ , \quad h_1 \ , \ \ldots \ , \quad h_n$$

where $e(g,g)^{\alpha}$ can be computed as $e(g^a, g^{a^n})^{y_0}$. $\mathcal{B}$ sends PK to $\mathcal{A}$.

Private Key Queries $\mathcal{A}$ is allowed to query the private key only for indices $i \in [1,n] \setminus \tilde{S}$. To answer the query, $\mathcal{B}$ generates $z_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and formally sets $r_i \leftarrow z_i - y_0 \cdot a^{n+1-i}$. It outputs

$$d_i \leftarrow \langle d_{i,0}, \ldots, d_{i,n} \rangle \quad \text{where} \quad d_{i,0} \leftarrow g^{-r_i} \ , \quad d_{i,i} \leftarrow g^{\alpha} h_i^{r_i} \ , \quad \forall_{j \neq i} \ d_{i,j} \leftarrow h_j^{r_i}$$

Notice that $\mathcal{B}$ can compute all these terms from the instance; in particular

$$d_{i,i} = g^{\alpha} h_i^{r_i} = g^{y_0 \cdot a^{n+1} + (y_i + a^i)(z_i - y_0 \cdot a^{n+1-i})}$$

which can be computed since the a^{n+1} term in the exponent cancels out.

Challenge $\mathcal{A}$ chooses a subset $S^* \subset \tilde{S}$. $\mathcal{B}$ sets

$$\text{Hdr} \leftarrow \langle C_1, C_2 \rangle \quad \text{where} \quad C_1 \leftarrow g^s \ , \quad C_2 \leftarrow (\prod_{j \in S^*} h_j)^s$$

It sets $K \leftarrow Z^{y_0}$. It sends $\langle \text{Hdr}, K \rangle$ to $\mathcal{A}$.

Notice that $\mathcal{B}$ can compute these terms from the instance. C_1 and K come directly from the instance. $\mathcal{B}$ can compute C_2 since it knows $\text{DL}_g(h_i)$ for all $i \in S^*$; in particular,

$$C_2 = (\prod_{j \in S^*} h_j)^s = (\prod_{j \in S^*} g^{y_j})^s = (g^s)^{\sum_{j \in S^*} y_j}$$

Guess Eventually, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ sends b' to the challenger.

Perfect Simulation From $\mathcal{A}$'s perspective, $\mathcal{B}$'s simulation has exactly the same distribution as the semi-static game defined in Section 2.2. The public and private keys are appropriately distributed, since α and the values $\{\mathrm{DL}_g(h_i)\}$ and $\{r_i\}$ are uniformly random and independent.

When $b = 0$ in the semi-static game, $\langle \mathrm{Hdr}, K \rangle$ is generated according to the same distribution as in the real world. This is also true in $\mathcal{B}$'s simulation: when $b = 0$, $K = e(g, g)^{\alpha \cdot s}$, and so the challenge is valid ciphertext under randomness s. When $b = 1$ in the semi-static game, $\langle \mathrm{Hdr}, K' \rangle$ is generated as in the real world, but K' is replaced by $K \xleftarrow{R} \mathcal{K}$, and $\langle \mathrm{Hdr}, K \rangle$ is sent to the adversary. This distribution is identical to that of $\mathcal{B}$'s simulation, where Hdr is valid for randomness s, but $K = Z$ is a uniformly random element of $\mathbb{G}_T$.

From this, we see that $\mathcal{B}$'s advantage in deciding the BDHE instance is precisely $\mathcal{A}$'s advantage against BE_{SS}.

B Proof of Theorems 3 and 4

First, a lemma. Let $p(x)q(x)|i$ denote the coefficient of x^i in $p(x)q(x)$.

Lemma 1. *Let $f_1(x), f_2(x) \in \mathbb{F}_p[x]$ be polynomials of degrees d_1 and d_2, respectively, whose resultant is nonzero. Let $d_3 \leftarrow d_1 + d_2 - 1$ and $i \in \{d_1, \ldots, d_3\}$. There exists a polynomial $t(x) \in \mathbb{F}_p[x]$ of degree d_3 such that $t(x)f_1(x)|_i = 1$, $t(x)f_1(x)|_j = 0$ for $j \in \{d_1, \ldots, d_3\} \setminus \{i\}$, and $t(x)f_2(x)|_j = 0$ for $j \in \{d_2, \ldots, d_3\}$.*

Proof. (Lemma 1) Consider the Sylvester matrix S of $f_1(x)$ and $f_2(x)$. The condition on $t(x)$ is equivalent to $S \cdot (t_0, \ldots, t_{d_3})^T = (0, \ldots, 0, 1, 0, \ldots, 0)^T$, where $t_i = t(x)|_i$. Since the resultant of $f_1(x)$ and $f_2(x)$ is nonzero, the Sylvester matrix is invertible. Set $(t_0, \ldots, t_{d_3})^T \leftarrow S^{-1} \cdot (0, \ldots, 0, 1, 0, \ldots, 0)^T$ and $t(x) = \sum_i t_i x^i$.

The complexity of computing $t(x)$ is $\mathcal{O}(d_2(d_1 + d_2))$ arithmetic operations over $\mathbb{Z}_p$.

Proof. (Theorems 3 and 4) This is given in the full version.

Traitors Collaborating in Public: Pirates 2.0

Olivier Billet[1] and Duong Hieu Phan[2]

[1] Orange Labs, Issy-les-Moulineaux, France
[2] Université Paris 8, Saint-Denis, France
olivier.billet@orange-ftgroup.com, hieu.phan@univ-paris8.fr

Abstract. This work introduces a new concept of attack against traitor tracing schemes. We call attacks of this type Pirates 2.0 attacks as they result from traitors collaborating together *in a public way*. In other words, traitors do not secretly collude but display part of their secret keys in a public place; pirate decoders are then built from this public information. The distinguishing property of Pirates 2.0 attacks is that traitors only contribute partial information about their secret key material which suffices to produce (possibly imperfect) pirate decoders while allowing them to remain anonymous. The side-effect is that traitors can publish their contributed information without the risk of being traced; giving such strong incentives to some of the legitimate users to become traitors allows coalitions to attain very large sizes that were deemed unrealistic in some previously considered models of coalitions.

This paper proposes a generic model for this new threat, that we use to assess the security of some of the most famous traitor tracing schemes. We exhibit several Pirates 2.0 attacks against these schemes, providing new theoretical insights with respect to their security. We also describe practical attacks against various instances of these schemes. Eventually, we discuss possible variations on the Pirates 2.0 theme.

1 Introduction

Traitor tracing is a cryptographic primitive introduced by Chor, Fiat, and Naor in [9] in the context of secure content distribution. This context covers for instance multimedia content rental, or broadcasting to a very large number of subscribers like in pay-TV systems, mass distribution of high value DVDs, or in web-based distribution of various multimedia contents. In all of these settings, the content is encrypted before its distribution in order to prevent illegal access which helps ensuring the revenues of the distributor. To decrypt the content, every legitimate user is provided with a decryption means, commonly called decoder. The main issue faced by the distributor is the construction and dissemination of unauthorized decoders, possibly creating a parallel market.

Hardware tamper resistant solutions are often too expensive compared to the price of the offered services. Furthermore, it would not prevent an organization from breaking into one box and extracting the necessary information to build and resell unauthorized decoders.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 189–205, 2009.

This is where traitor tracing schemes step into the game: the key material embedded in the decoders is diversified on a user basis. Thus, decoders are 'marked' with the identity of the user and traitor tracing allows the authority to trace a user that produced a pirate decoder. Such users, called traitors, are more powerful when they collude to create a pirate decoder. In this case, traitor tracing should allow the tracing of at least one of the traitors that took part in the coalition. A trivial solution to the problem of traitor tracing is to provide every user with a randomly chosen key that identifies him and encrypt the content as many times as there are users in the system. Obviously, such a solution is totally impracticable due to bandwidth restrictions. Hence, bandwidth preservation in traitor tracing schemes is of crucial importance.

Since the seminal work of Chor, Fiat, and Naor, there have been several proposals and improvements in traitor tracing schemes. We give a few landmarks of the work in traitor tracing but this list is of course not exhaustive. Boneh and Franklin exposed an elegant algebraic construction coming with a deterministic tracing procedure in [4]. Fiat and Tassa proposed a way to dynamically remove traitors from the system once they are caught, see [12]. Kiayias and Yung gave a powerful method to turn black-box tracing against stateless decoders into black-box tracing against stateful decoders in [16]. In [6], Boneh, Sahai, and Waters introduced a full collusion traitor tracing scheme with sub-linear ciphertext size and constant size private keys. Traitor tracing schemes based on codes have been much investigated since the seminal work of Boneh and Shaw [7]: Kiayias and Yung [17] proposed a scheme with constant rate, [8,19] relaxed the assumption that the tracer is a trusted third party, and [3,5] recently achieved constant size ciphertexts. Among the most famous traitor tracing schemes are schemes from the NNL framework [18] as they were used as a basis to design the widely spread content protection system for HD-DVDs and Blu-ray disks called AACS [1]. These are not exactly traitor tracing schemes, but rather very efficient broadcast encryption schemes with some black-box tracing abilities.

Pirates 2.0 attacks are primarily targeted to code based schemes and schemes from the NNL framework, but might be used against other combinatoric schemes.

1.1 Collaborative Traitors: Pirates 2.0

From the point of view of the attack model for traitor tracing schemes, there has been no radical change since the introduction of the concept in [9]. One remarkable exception is Pirate Evolution from [15] which exposes a new threat against trace and revoke schemes such as [18]. In this paper, we introduce another new threat that we call Pirates 2.0 against both traitor tracing schemes and the trace and revoke schemes from [18].

The main characteristics of our new Pirates 2.0 threat are as follows:

Anonymity Guarantee: Traitors that participate in a Pirates 2.0 attack are provided with a guarantee (through the exhibition of a mathematical proof) that *they cannot be traced by the authority.*

Partial Contributions: Traitors never need to reveal their whole secret key.

Public Collusions: Traitors operate in a public environment: they publish secret data from their decoders.
Large Coalitions: Traitors take part in unusually large coalitions.
Dynamic Coalitions: Traitors can come into action only when necessary.

The anonymity guarantee together with considerations on imperfect decoders makes the basis of our attack scenario and everything else heavily relies on it. The anonymity guarantee indeed gives strong incentives to potential traitors to actually take the plunge: With ubiquitous access to the Internet, leaking secret data, say, in a peer-to-peer network without further action can be done very quickly and in a straightforward way. This makes it an appealing scenario among the ever growing [11,10,24] set of users hostile to the currently deployed Digital Rights Management systems (DRM). The characteristic that large coalitions can easily be achieved is therefore a direct consequence of the fact that traitors are *guaranteed not to be traced* by an authority.

Considerations on imperfect decoders are the other determinant ingredient: A pirate decoder is considered to be useful if it can decrypt (resp. decrypt with a high probability) valid ciphertexts; such a pirate decoder is called perfect (resp. imperfect) decoder. In previous work, it is assumed that a pirate decoder always decrypts ciphertexts from the tracer when it is not able to detect the presence of the tracing procedure, i.e. it is assumed that the pirate decoder is either perfect or only slightly imperfect. This assumption makes sense in the classical model of coalitions since any coalition, knowing at least one legitimate key, is able to decrypt all valid ciphertexts anyway. However, in the Pirates 2.0 setting, we show that another trade-off is possible for the pirates when the scheme uses variable length ciphertexts: the pirate decoder is only required to decrypt ciphertexts reasonably shaped. As an example of this scenario, consider the NNL scheme where the expansion of valid ciphertexts can vary a lot: A pirate decoder that can decrypt ciphertexts of size lower than, say 1 GB, is *highly* imperfect, but still useful to pirates.

In this paper, we show that some traitor tracing schemes and trace and revoke schemes (including the NNL scheme from [18] and code based schemes) are susceptible to Pirates 2.0 attacks. We give several practical attacks against various instances of such schemes, most notably against the AACS. We then derive the theoretical implications for all these traitor tracing schemes.

1.2 Comparing Pirates 2.0 and the Classical Setting

We summarize below the main differences between our new attack model and the classical one:

Motivation: The classical model for coalitions captures the fact that pirates might invest some amount of money in order to sell unauthorized decoder to the black market. In the case of Pirates 2.0, the motivation might be to get rid of a protection system to which a large number of users are hostile. In the history of DVDs for instance, the main motivation to crack the system came from compatibility issues: the protection was thought to be too restrictive.

Static vs. Adaptive: The classical model of pirates is static. The coalitions consist of randomly chosen decoders. Therefore it is not possible to bias the collection process. In a Pirates 2.0 attack, traitors are able to contribute information adaptively, that is, depending on the current state of affair at the moment of the contribution. Therefore, even if during the publication process each traitor operates isolated (i.e. without communication with the other traitors), having access to published information at the time of the contribution makes it a collaborative process.

Anonymity: In the classical model of coalitions, traitors colluding must trust each other, or at least, one third party (say the pirate who collects the secret data). In contrast, the Pirates 2.0 attack only requires that the partial secret information provided by the traitors guarantees their anonymity.

Size of Coalitions: In the classical models, one usually assumes a small number of traitors (especially for combinatorial schemes like those relying on codes or those based on trees). This assumption seems reasonable in the classical model because each traitor must trust a third party and even in the case of an isolated traitor, getting a large number of decoder legally might be very expensive. In Pirates 2.0, this assumption becomes wrong, since the traitors guaranteed to remain anonymous can form a very large coalition.

2 Formalization of Pirates 2.0

There are many possible settings for a Pirates 2.0 attack. For instance, the construction of a pirate decoder can be active or passive. In the active case, the contributions made by the traitors are driven by the pirate upon building the pirate decoder. In the passive case, the traitors contribute information at their discretion. In this work, we focus on the last of these scenarios which leaves more freedom to the traitors and makes the attack even more realistic.

Also, there are two possible ways of collecting the information contributed by the traitors: in a centralized way or in a distributed way. Again, the distributed way leads to a stronger attack with less constraints in practice: traitors can easily use peer-to-peer networks to contribute their information, whereas a centralized server is more susceptible to shut down by legal action. We therefore choose to focus on the distributed setting, though in some cases, assuming a centralized entity like a pirate server would render the work of contributing for the traitors and of building a pirate decoder easier than in a peer-to-peer network. In the rest of the paper, we point out where it is relevant to use the facilities that a pirate server would provide.

2.1 A Setting for Pirates 2.0

We now describe several concepts that we use in the Pirates 2.0 setting:

Traitors and Pirates. As usual, a traitor is a legitimate user in possession of some secret data that we call his secret key and who leaks part of this secret key. Pirates are not legitimate users: they are not entitled to secret data but are able

to collect relevant information from their public environment in order to produce a pirate decoder. We naturally assume that pirates and traitors respectively collect and contribute information in a stateful way: a traitor keeps track of (all) the information he contributed to the public, whereas both pirates and traitors can keep track of all of the information that was contributed to the public.

Contributed Information. The contributed information is the sum of information that was put into the public domain by the traitors at a given point in time, i.e. the secret data leaked from the system. The current contributed information at any point in time is denoted by $\mathcal{C}$. Initially, $\mathcal{C} = \emptyset$.

Traitor's Strategy. A traitor's strategy is a publicly available probabilistic algorithm Contribute that traitors execute to provide information to pirates. A traitor's strategy comes with a certificate that information leaked following this strategy allows the traitors to preserve some anonymity level. Traitors might in principle use different strategies, but for simplicity we only consider in the following the case where all traitors implement the same strategy.

The strategy Contribute, takes as input the traitor's secret key sk, some information I already contributed by other traitors (for instance the set $\mathcal{C}$ of all contributed information at the time Contribute is run) as well as the history H of the contributions made by the traitor. The traitor's strategy returns $\mathsf{Contribute}(sk, I, H)$ as the traitor's contribution to the public. (And therefore, the overall information contributed to the public $\mathcal{C}$ is accordingly updated: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathsf{Contribute}(sk, I, H)$.)

Public Information. The public information $\mathcal{P}$ consists of all the public data available from the broadcaster (such as for instance its public key, the public key of users if any, etc.) together with the contributed information $\mathcal{C}$.

Anonymity Level. The public procedure Anonymity provides the level of anonymity $\mathsf{Anonymity}(sk, S, \mathcal{P})$ of a traitor with the secret key sk who leaked an information S (which corresponds to the sum of all his contributions) following a public strategy (we refine this notion later on by using extraction functions). The anonymity level output by the procedure corresponds to the uncertainty on the traitor's identity from the tracing authority point of view when provided with the sequence of contributed information S. At level 1 the traitor is known, while at level N, the traitor is undistinguishable from another user.

Pirate Decoder. We think of a pirate decoder as the output of an algorithm called Pirate. If the amount of information available from $\mathcal{P}$ is large enough, Pirate produces a pirate decoder $\mathsf{Pirate}(\mathcal{P})$ and simply outputs '`failed`' otherwise.

In the following we assume that the contribution of secret data to the public domain $\mathcal{C}$ by the traitors is a discrete process.

Definition 1 (Security against Pirates 2.0). *A traitor tracing scheme is said to be $\alpha-$secure against Pirates 2.0 if it prevents the construction of pirate decoders from information published by traitors with an anonymity level greater than α.*

Note that not all traitor tracing (or trace and revoke) schemes are susceptible to Pirates 2.0 attacks. On the other hand, even fully collusion resistant schemes might be at risk as Pirates 2.0 attacks allow highly imperfect decoders: decoder

can refuse to decrypt classes of specific ciphertexts—e.g. depending on their size. As we will show in the next sections, some of the most famous schemes, including the one used in the AACS, are susceptible to our new attack strategy.

2.2 A Concrete Treatment of Anonymity Estimation

The basic idea behind Pirates 2.0 attacks is that traitors are free to contribute some piece of secret data as long as several users of the system could have contributed exactly the same information *following the same (public) strategy*: this way, they are able to remain somewhat anonymous. The anonymity level is meant to measure exactly how anonymous they remain.

Definition 2 (Extraction Function). *An extraction function is an efficiently computable function f that outputs information about the secret key.*

Definition 3 (Masked Traitor). *A traitor t is said to be masked by a user u for an extraction function f if $f(sk_u) = f(sk_t)$.*

This notion of a traitor being masked by another user in the system is the basic undistinguishability notion that allows us to estimate the level of anonymity of a traitor after his contribution:

Definition 4 (Anonymity Level). *The level of anonymity of a traitor t after a contribution $\cup_{1\leq i\leq n} f_i(sk_t)$ is defined as the number α of users masking t for each of the n extraction functions f_i simultaneously:*

$$\alpha = \#\{u \mid \forall i,\ f_i(sk_t) = f_i(sk_u)\}\ .$$

In the previous definitions, we use the equality between each extraction function f_i to derive the anonymity level. One can wonder why not simply consider equality between the *global* information leaked by a traitor and the global information another user u could extract like $\cup_i f_i(sk_t) = \cup_j g_j(sk_u)$ with any set of extraction functions $\{g_j\}$. The answer is that we do not want to keep the traitor strategy secret and therefore, the authority can, at least from a theoretical point of view, use its knowledge of the set of extraction functions $\{f_i\}$ used by the traitors to gain additional information and to trace the traitors. (It might well be that there exists another user u such that $\cup_i f_i(sk_t) = \cup_j g_j(sk_u)$ holds, but $\cup_i f_i(sk_t) = \cup_i f_i(sk_v)$ would have been impossible for any user v other than t.)

3 Pirates 2.0 and the Subset-Cover Framework

The subset-cover framework proposed by Naor, Naor, and Lotspiech in [18] is a powerful tool to design efficient trace and revoke systems. It captures many previously proposed traitor tracing systems and forms the basis of the so called NNL scheme used in the content protection system for HD-DVDs known as AACS [1]. However, as we show in this section, this scheme is susceptible to our attack and we explain how to defeat the AACS system.

3.1 Brief Description of the Subset-Cover Framework

The subset-cover framework is a powerful means to capture several trace and revoke designs. It encompasses several traitor tracing schemes proposed to date and maybe even more importantly, serves as the basis for two of the most efficient trace and revoke schemes: the complete subtree scheme and the subset difference scheme.

In the subset-cover framework, the set $\mathtt{N}$ of users in the system is covered by a collection of subsets $\mathtt{S}_i$ such that $\cup_i \mathtt{S}_i \supset \mathtt{N}$ and $\mathtt{S}_i \cap \mathtt{N} \neq \emptyset$. This covering is not a partition of $\mathtt{N}$ and the sets $\mathtt{S}_i$ rather overlap. To every subset $\mathtt{S}_i$ corresponds a long term secret key L_i, and every user that belongs to $\mathtt{S}_i$ is provided with this secret key—or in an equivalent way, with some material that allows him to derive this secret key. Therefore, every user u of the system is given a collection of long term keys $\{L_{i_k}\}$ that together form his secret key which we denote by sk_u.

In order to broadcast some content M, the center uses a standard hybrid scheme: a session key K is first drawn randomly and used to encrypt (with an encryption scheme E') the content, before being encrypted under multiple long term keys (with another encryption scheme E). The long term keys L_{i_k}, $k = 1, \ldots, l$ are chosen so that the corresponding subsets $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_l}$ only cover the set of users entitled to decrypt. Therefore, the center broadcasts ciphertexts of the form:

$$\Big[\big(i_1, E_{L_{i_1}}(K)\big), \big(i_2, E_{L_{i_2}}(K)\big), \ldots, \big(i_l, E_{L_{i_l}}(K)\big) \parallel E'_K(M) \Big]$$

To decrypt, a valid decoder for user u performs the following sequence of operations: It first looks for an index i_j in the first element of each of the l couples $(i_k, E_{i_k}(K))$ in turn such that $\mathtt{S}_{i_j} \subset sk_u$. If no index correspond, the decoder does not decrypt; otherwise, the decoder retrieves the corresponding long term key L_{i_j} and uses it to decrypt the associated encrypted session key $E_{i_j}(K)$ and then decrypts the payload $E'_K(M)$.

Since the system is built to handle revoked users, let us also denote by $\mathtt{R}$ the set of revoked users in the system at any point in time. In order to prevent them (independently, but also working together as a coalition) from accessing the encrypted content $E'_K(M)$, the collection $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_l}$ is specially crafted so that:

$$\bigcup_{k=1}^{l} \mathtt{S}_{i_k} = \mathtt{N} \setminus \mathtt{R} \ .$$

The tracing procedure. Now that we showed how the system deals with revoked users, we have to describe the way it disables pirate decoders. As is usual, the tracing procedure works with black-box access to the pirate decoder only. The idea is to refine the covering initially used to broadcast ciphertexts so that the pirate decoder cannot decrypt with probability p higher than some threshold. To this end, the authors of [18] suggest to use an hybrid argument: the pirate box is provided with "ciphertexts" with payload $E'_K(M)$ and headers of type j (for $j = 1, \ldots, l$):

$$\big(i_1, E_{L_{i_1}}(R)\big), \ldots, \big(i_j, E_{L_{i_j}}(R)\big), \big(i_{j+1}, E_{L_{i_{j+1}}}(K)\big), \ldots, \big(i_l, E_{L_{i_l}}(K)\big)$$

where R is some randomly chosen element independent from K. If we denote by p_j the probability that the pirate box correctly decrypts the specially crafted ciphertexts of type j, there must exist an index t such that $|p_t - p_{t-1}| \geq \frac{p}{l}$ and therefore some traitor belongs to $\mathtt{S}_{i_t}$. The tracer then iterates this basic procedure, applying it to an arbitrary covering of $\mathtt{S}_{i_t}$ until either $\mathtt{S}_{i_t}$ contains a single element (which thus matches a traitor) or the pirate box cannot decrypt above the threshold (and no one is accused of being a traitor, but the new partition renders the pirate box useless).

The authors of [18] showed that this tracing procedure is correct as soon as the revocation scheme satisfies a so-called "bifurcation property": every subset can be split into two subsets of roughly the same size. As we will see, this is the case for the two schemes *complete subtree* and *subset difference*.

3.2 General Attack Strategy against Subset-Cover Schemes

The generic process for the attack is relatively simple and runs in a few steps:

Elaborating the strategy

The main idea is to select a collection of subsets $\mathtt{S}_{\iota_1}, \ldots, \mathtt{S}_{\iota_w}$ such that:

- The number of users in each subset $\mathtt{S}_{\iota_k}$ is large, so that the anonymity level of the traitors is guaranteed to remain high enough when they contribute the associated long term key L_{ι_k};
- For any set $\mathtt{R}$ of revoked users and any method used by the broadcaster to partition $\mathtt{N} \setminus \mathtt{R}$ into subsets $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_m}$, the probability that one of the subsets $\mathtt{S}_{\iota_k}$ belongs to the partition $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_m}$ is high—say exceeds a given threshold τ—or the broadcaster exceeds its available bandwidth.

Contributing data

Let us define the extraction functions f_i to be $f_i(sk) = L_i$ if $L_i \in sk$ and '`missing`' otherwise. To contribute part of his private key sk_t, a traitor t performs the following sequence of lookups: for each index i from $\{\iota_1, \iota_2, \ldots, \iota_w\}$ (taken in any order) the traitor computes $C = f_i(sk_t)$ and if $C \neq \mathtt{failed}$ and $C \notin \mathcal{P}$ returns and outputs C. The information H about sk_t that the traitor already contributed to the public is included in the argument list so that the contribution is $\mathsf{Contribute}(sk_t, \mathcal{P}, H)$.

Building pirate decoders

A pirate decoder simply embeds the public keys $L_{\iota_1}, \ldots, L_{\iota_w}$. Upon reception of a ciphertext

$$\Big[\big(i_1, E_{L_{i_1}}(K)\big), \big(i_2, E_{L_{i_2}}(K)\big), \ldots, \big(i_l, E_{L_{i_l}}(K)\big) \parallel E'_K(M) \Big]$$

from the center, the pirate checks whether $\{\iota_1, \ldots, \iota_w\} \cap \{i_1, \ldots, i_m\} = \emptyset$. If not, that is if there is an index $\iota_k = i_l$ in both sets (which was assumed to occur with high probability), the pirate box recovers the corresponding key L_{ι_k}, uses it to decrypt the session key K from $E_{L_{i_l}}(K)$, and therefore is able to correctly decrypt the payload.

Anonymity

The level of anonymity of a given traitor t in a subset cover scheme is related to the number of users of the system that know the complete list of subsets $\mathtt{S}_{t_1}, \ldots, \mathtt{S}_{t_l}$ for which the traitor contributed the keys $L_{t_1}, \ldots, L_{t_l}$ to the public.

3.3 Pirates 2.0 against the Complete Subtree Scheme

The complete subtree scheme. In this scheme, the users correspond to the leaves of a complete binary tree whereas the collection of subsets $\mathtt{S}_i$ exactly corresponds to all the possible subtrees in the complete tree. When $|\mathtt{N}| = 2^n$, the complete binary tree is of length n and there are exactly n subtrees that contain a given leaf. Figure 1 shows a covering using six subsets of twelve users that excludes four revoked users (depicted in black). This subset scheme complies with the bifurcation property since any subset (or equivalently any subtree of the complete binary tree) can be split into two subsets of equal size (the two subtrees rooted at the two children of the root of the original subtree). Regarding key assignment, each user represented by a leaf u in the complete binary tree is provided with the keys L_i associated to the nodes i on the path from the leaf u to the root.

Covering algorithm. In the case of the complete subtree, the covering used to exclude the $r = |\mathtt{R}|$ revoked users from $\mathtt{N}$ is the collection of subsets that hang off the Steiner tree of the revoked leaves. (The Steiner tree of the revoked leaves is the minimal subtree of the complete binary tree that connects all the revoked leaves to the root and it is unique.) Since any user only knows the keys from its leaf to the root and since this path is included in the Steiner tree for revoked users, these users cannot decrypt anymore. This algorithm produces covers of size $O(r \log(N/r))$.

We now give a version of our attack against subset cover schemes in the case of the complete subtree scheme:

Theorem 1. *On average, a randomly chosen group of $\rho \log \rho$ traitors (operating isolated) is able to mount a Pirates 2.0 attack against a complete subtree scheme*

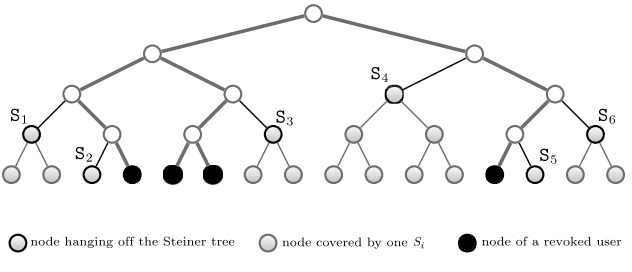


Fig. 1. Complete subtree: leaves correspond to users, $\mathtt{S}_1, \ldots, \mathtt{S}_6$ is the covering that excludes revoked users in black while allowing other users to decrypt derived from the Steiner tree associated to the set of revoked users $\mathtt{R}$

in which the center wants to ensure a ciphertext rate[1] *of at most* $\rho(N-r)/N$. *Moreover, each traitor is guaranteed an anonymity level of* N/ρ.

Proof. For simplicity we assume that no collision occurs during the contribution process (the traitors contribute sequentially, although in a completely random way, their share of secret data) and that the contribution of a traitor is readily available to the public. (It is obviously possible to deal with these refinements by considering statistical processes instead and then bounding the loss in efficiency that would occur in such a general case.)

Following the general attack strategy described in the previous section, define $\mathtt{S}_{\iota_1}, \ldots, \mathtt{S}_{\iota_w}$ to be the subsets corresponding to all the subtrees of the complete tree having more than N/ρ leaves so that for each ι_k more than N/ρ users share the corresponding long term keys L_{ι_k}. These subsets also correspond to all the nodes between level 0 (the root) and the level $\lambda = \lfloor \log \rho \rfloor$ and thus, there are $w = 2\lfloor \rho \rfloor$ of them. Then, a traitor contributing one of the L_{ι_k} at level λ together with every L_{ι_j} on the path from node ι_k to the root has a level of anonymity[2] higher than N/ρ. (As mentioned above, more than N/ρ users share the key L_{ι_k} and moreover the same users also know about L_{ι_i} for every node ι_i on the path from node ι_k to the root because of the assignment scheme.) Now, the number of traitors needed to collect the $\lfloor \rho \rfloor$ long term keys (and those above) is given by the answer to the classical coupon collection problem: to collect all the m possible items when one receives a uniformly chosen item at each draw requires $m \log m$ draws on average. This demonstrates the first part of the theorem.

It only remains to show that either a pirate is able to produce a working decoder, or the center uses too much bandwidth (the ciphertext rate is bigger than ρ). Let r be the number of revoked users. Let us assume that the broadcaster only uses subsets rooted at a level $l \geq \lambda$ since otherwise the priate decoder is able to decrypt the ciphertexts. Now every subset can cover at most $N/2^{\lambda}$ users so that $\rho(N-r)/N$ of them are needed to cover the $N-r$ legitimate users. □

Theoretical and practical impact. From a theoretical point of view, Theorem 1 shows that instead of the $O(r \log(N/r))$ complexity that was first derived, the bandwidth required for the complete subtree scheme to operate securely actually is $O(\rho(N-r)/N + r \log(N/r))$ for a number of $\rho \log \rho$ traitors taking part in a Pirates 2.0 attack.

From a practical point of view, we note that we assumed that every long term key can be leaked by at least one traitor. For a system accommodating 2^{32} users and a long term key at the 12th level, this assumption translates into the fact that among a million of users there is at least one that takes the step of contributing it to the public (with the guarantee of remaining anonymous!); this hypothesis seems reasonable to us.

[1] The ciphertext rate is the number of subsets used by the center.

[2] Having a lot of revoked users in the subtree does not affect the level of anonymity: revoked users know the keys on their path to the root and could have contributed them as well. This, however, affects the decryption threshold of the pirate decoder.

Also, note that even in the case where one long term key is not contributed by any user, the attack remains valid: the pirate box will not be able to decrypt only with a very small probability.

3.4 Pirates 2.0 against the Subset Difference Scheme

The subset difference scheme has been introduced to lower the number of subsets required to partition the set of legitimate users $\mathtt{N} \setminus \mathtt{R}$. It improves on the complete subtree scheme exposed above by a factor of $\log(N/r)$ in terms of bandwidth usage for the headers.

To attain this level of performance, the number of possible subsets has been tremendously increased. Remember that $\mathtt{S}_i$ denotes the full binary subtree of the complete binary tree rooted at node i. Now, for each node j in $\mathtt{S}_i$ different from i, let us denote by $\mathtt{S}_{i,j}$ the binary subtree rooted at node i of which the full binary subtree rooted at node j has been removed. (See examples in Figure 3.) A user will need to know all the keys $L_{i,j}$ such that he belongs to the subtree rooted at i but not in the subtree rooted at j. However, it would be impossible for each device to store such a huge number of long term keys. This is why a key derivation procedure has been designed to allow the derivation of most of the $O(N)$ long term keys a user is entitled from a much smaller set of keys: a user only needs to store $O(\log^2(N))$ keys. Each node i in the full binary tree is first assigned a random label LABEL_i and labels $\mathsf{LABEL}_{i,j}$ together with their corresponding long term keys $L_{i,j}$ are deduced (in a pseudo-random way) from label LABEL_i. The key derivation procedure then works as follows: from each LABEL_i, a pseudo-random value $\mathsf{LABEL}_{i,j}$ is obtained for each sub-node j using the tree based construction proposed by Goldreich, Goldwasser, and Micali [13]; from this value, a long term key $L_{i,j}$ is eventually deduced (in a pseudo-random way). Each user is then provided with labels $\mathsf{LABEL}_{i,j}$ for all nodes i that are on the path from the leaf that represents the user to the root, and all nodes j hanging off this path as described on Fig. 2. This key assignment ensures that every user in the subtree rooted at node i but not in the subtree rooted at node j is able to derive $L_{i,j}$ while every user in the subtree rooted at node j is not able to derive $L_{i,j}$.

Covering algorithm. The covering algorithm works by maintaining a subtree T of the Steiner tree of $\mathtt{R}$ and removes nodes from it at each steps:

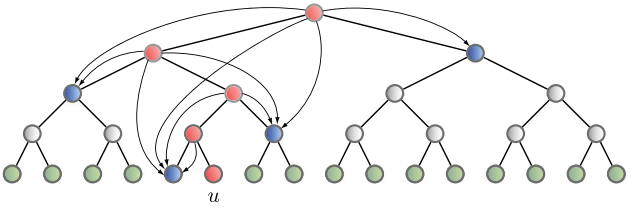


Fig. 2. Key assignment. User u receives all the labels $\mathsf{LABEL}_{i,j}$ such that i is a parent of j and i is on the path from the leaf of u to the root.

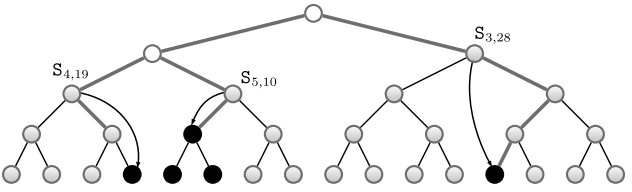


Fig. 3. Subset difference: leaves correspond to users and black nodes are not able to derive the necessary information to decrypt. Therefore $\mathtt{S}_{4,19}$ prevents user 19 from decrypting, $\mathtt{S}_{5,10}$ prevents users 20 and 21 from decrypting, and $\mathtt{S}_{3,28}$ prevents user 28 from decrypting. All other users are able to decrypt.

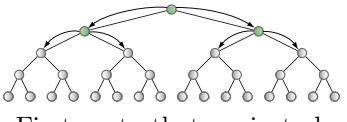
`init:` Make T the Steiner tree of $\mathtt{R}$.

`select:` If there is only one leaf v_k in T and it is not the root (or node 0), add the subset $\mathtt{S}_{0,k}$ and return. If there is only the root in T, return. Otherwise, select two leaves v_{j_1} and v_{j_2} from T so that their least common ancestor v does not contain any other leave of T than v_{j_1} and v_{j_2}. Call v_{i_1} and v_{i_2} the children of v such that v_{i_1} is the ancestor of v_{j_1} and v_{i_2} the ancestor of v_{j_2}. Then, if $v_{i_1} \neq v_{j_1}$ add $\mathtt{S}_{i_1,j_1}$ to the partition and similarly if $v_{i_2} \neq v_{j_2}$ add $\mathtt{S}_{i_2,j_2}$ to the partition. Remove all the descendants of v from T, which makes v a leaf of T. Reiterate the step 'select'.

An example output of this procedure is shown in Figure 3.

Theorem 2. *On average, a randomly chosen group of $\rho \log \rho$ traitors (operating isolated) is able to mount a Pirates 2.0 attack against a subset difference scheme in which the center wants to ensure a ciphertext rate of at most $\rho(N-r)/N$. Moreover, each traitor is guaranteed a level of anonymity of at least $N/2\rho$.*

Proof. In the following proof we make use of labels of a special type, that we call *direct labels*. Direct labels are $\mathsf{LABEL}_{i,j}$ such that the node j is a *direct* descendant of the node i. The first six direct labels of the tree are described in the figure on the left.



First, note that a pirate knowing all the keys $L_{i,j}$ where the node i lies in the first $\lambda = \lfloor \log \frac{\rho}{2} \rfloor$ levels, is able to decrypt all the ciphertexts where the rate is lower than $\rho(N-r)/N$ where r is the number of revoked users. Indeed, the broadcaster must use subsets $S_{k,l}$ where the node k does not lie in the first λ levels in order to prevent the pirate from decrypting the ciphertexts. Since each of these subsets covers less than $N/2^{\lambda+1}$ users (those who are in the subtree rooted at node k), the center must use at least $\rho(N-r)/N$ subsets to cover all the legitimate users.

Collecting all the keys rooted at a level $l \leq \lambda$ is however totally unpractical since there are a tremendous number of such keys. The pirate can nevertheless go around this difficulty by collecting labels $\mathsf{LABEL}_{i,j}$ instead of keys $L_{i,j}$ and using the derivation procedure to lower the minimum information to be kept: the labels that users possess allow to derive a large number of keys. Therefore,

we claim that it is enough for the pirate to collect all *direct* labels $\mathsf{LABEL}_{i,j}$ where i is located in the first λ levels in order to derive all keys $L_{i,k}$. (Once the pirate knows the two direct labels at node i, he can derive all keys $L_{i,k}$ where k is in the subtree rooted at i.)

To prove the theorem, we show that on average, $\rho \log \rho$ randomly chosen traitors are able to contribute all the direct labels of the first λ levels. Each traitor contributes all his direct labels $\mathsf{LABEL}_{i,j}$ for the nodes i located in the first λ levels. Note that at each level, a traitor has been assigned exactly one of the direct labels. Thus, when all direct labels at level exactly λ have been contributed, so have the direct labels of all the first $\lambda - 1$ levels. As a randomly chosen traitor knows a uniformly chosen direct label out of the $\frac{\rho}{2}$ direct labels of level λ, a randomly chosen group of $\rho \log \rho$ traitors (operating isolated) is able to contribute all direct labels $\mathsf{LABEL}_{i,j}$ where i is located in the first λ levels.

Moreover, such traitors share their contribution with every user in the same subtree rooted at level $\lambda + 1$: each traitor is covered by N/ρ users. □

Remark 1. Theorem 2 is proven in the case of static attacks: traitors submit information non-adaptively, such as in a peer-to-peer scenario. However, the number of required traitors to mount a Pirate 2.0 attack can be lowered to ρ in the case of an adaptive attack such as in a server-based scenario.

Impact on AACS. In the case of AACS, the subset difference scheme is used with $N = 2^{31}$ users. The header is written in a so-called Media Key Block or MKB for short which (among other) encodes the indices for the difference subsets as well as the media key encrypted once for each of the corresponding long term keys. These keys are 16 bytes long and the indices are encoded using 5 bytes. According to Section 3.2.5.5 of AACS specifications [2]: "*For the purposes of designing performance, a one megabyte buffer is sufficient to process the MKB.*" Although this is not an intrinsic limitation of the system, very large MKBs would decrease the performances of hardware devices and would increase their price. This is why applications like disk replicators often only allocate 1MB space for the MKB. In the case of AACS, this means that only $2^{11.6} = 2^{20}/21$ encrypted keys will be able to fit this space and thus a Pirates 2.0 attack against the AACS would only require some thousand collaborating traitors which, given the guarantee offered to traitors (a million of other users cover each traitor), seems very practicable.

Also note that once again the attack given here is just an illustration of our general concept of attack. There are several possible improvements and refinements such as taking advantage of the partition algorithm (remember that the scheme is a trace and revoke scheme and not a full traitor tracing scheme, so that it might fail to single out a traitor).

4 Pirates 2.0 and Code Based Schemes

Traitor tracing schemes based on codes (be it collusion secure codes [7,25] or identifiable parent property codes [14,22]) have been proposed during more than

half a decade [17,8,21,20,23,5]. Their main advantage is their efficiency in terms of bandwidth requirements, but their main drawback is that their efficiency (in terms of the size of the private key) is highly sensitive to the number of traitors in the coalition.

4.1 General Framework of Codes Based Schemes

Traitor tracing schemes built on codes more or less fit in the following framework:

Setup: The scheme generates a code $\mathtt{C}$ of length ℓ which is either a collusion secure code or a q-ary c-IPP code. The alphabet for the code is $\mathtt{A} = \{0, 1\}$ in the case of a collusion secure code and $\mathtt{A} = \{1, \ldots, q\}$ in the case of an IPP code. Then, for each position $i = 1, \ldots, \ell$ in a codeword and for each possible letter a from $\mathtt{A}$, a key $K_{i,a}$ is randomly chosen. Hence, there are 2ℓ possible keys (resp. $q\ell$ possible keys) in the system in the case of collusion secure codes (resp. IPP codes).

Key assignment: Each user u is given a codeword W_u from $\mathtt{C}$. Then, for each position $i = 1, \ldots, \ell$ in this codeword, the user is provided with the key $K_{i,W_u[i]}$ where $W_u[i]$ is the letter at position i in the codeword W_u. Thus, each user gets ℓ keys in its decoder.

Decoder: A ciphertext usually contains a header that specifies the positions of the keys involved in the decryption process. For instance, in the case of the scheme [17] proposed by Kiayias and Yung, all the keys of the user are involved. In the case of the scheme [5] proposed by Boneh and Naor only one key is involved during a decryption process.

4.2 Pirates 2.0 against Code Based Schemes

Our goal is to show how our generic attack can be applied to this class of schemes. We do not focus on any concrete construction but rather deal with the underlying codewords. For ease of exposition, we describe an attack when the underlying code is a Tardos' code [25] but this attack might easily translate to other codes.

First, recall that a Tardos' code secure against coalitions of size at most c is built as follows. First, the code length is set to be $\ell = \lceil 100c^2 \log(N/\epsilon) \rceil$. Then, for each integer i in the interval $[1, \ldots, \ell]$ a (secret) value $0 < p_i < 1$ heavily biased towards 0 or 1 is randomly drawn. Then, any of the N codewords is constructed by randomly choosing for each position i in $[1, \ldots, \ell]$ the bit '$\mathtt{0}$' or the bit '$\mathtt{1}$' according to the probability p_i.

Theorem 3. *For any traitor tracing scheme that relies on Tardos' code for its set of keys, a set of T traitors collaborating to mount a Pirates 2.0 attack allows to produce a pirate decoder while maintaining a level of anonymity higher than $N \cdot 2^{-\ell/T}$ on the average.*

Proof. Since contributing large amounts of a codeword makes your level of anonymity drop a lot, a strategy that handles every traitor with equity is to make them contribute the same amount of secret data. Since there are T traitors,

let them each contribute ℓ/T elements of (the secret data associated with) their codeword. Of course, people are then already able to construct pirate decoders with the collected material. The anonymity level α a traitor can expect is easy to assess: if $m = \lceil \ell/T \rceil$,

$$\alpha = N \prod_{i=1}^{m} \left(p_{\sigma(i)}^2 + (1 - p_{\sigma(i)})^2 \right) \quad . \tag{1}$$

Indeed, for a randomly chosen traitor, there is a probability p_i that the letter at position i is '0' and for any other codeword randomly chosen a probability p_i that the letter at that position is also '0'. Similarly there is a probability $1-p_i$ that the letter at position i is '1' and the same probability that another codeword gets the same letter at that position. Therefore, the probability that another codeword gets the same letter as that of the traitor for some position i is $q_i = p_i^2 + (1-p_i)^2$. The probability that a block of size m of the traitor's codeword is the same as that of another user is thus $\prod_{i=1}^{m} q_{\sigma(i)}$, where σ is a permutation of $\{1, \ldots, \ell\}$ that accounts for the particular selection of the block of size m.

The sum from Eq. (1) takes into account every possible block of codeword of length m and by multiplying by the total number of users in the system, we get the average number of users masking a randomly chosen traitor, that is its level of anonymity in the system. Now since $p_i^2 + (1 - p_i)^2 \geq \frac{1}{2}$ we get a (very loose) bound on the level of anonymity: $\alpha \geq N \cdot 2^{-\ell/T}$. □

Theoretical and practical impact. From a theoretical point of view, the above theorem shows that the number of traitors required to mount a Pirates 2.0 is only *linear in the size of the decoder* and only logarithmic in the number of users in the system. From a practical point of view, it would require about 2^{17} traitors to mount a Pirate 2.0 attack against a traitor tracing scheme that relies on a 30-collusion secure code with 2^{32} users. Each traitor would be masked by about a few thousand users in this case.

5 Conclusion

Throughout this paper we presented a novel concept of attack against combinatorial traitor tracing schemes. We focused on the main ideas behind this concept of attack, but some variations could be further investigated. For instance, it is possible to consider the case of dishonest traitors (a common threat to collaborative work is bad contributions which have to be tracked and eliminated). Dishonest traitors capture the fact that the authority could try to perturb the creation of pirate decoders by publishing incorrect information. However, one of the traitors might use its own authorized decoder to verify the contribution of the other traitors: after having sorted out these contributions, he is able to produce a pirate decoder.

Another direction is to consider probabilistic guarantees for the level of anonymity of contributing traitors: the traitors are only certified to have a high level of anonymity with some (possibly very high) probability. This is useful if

the authority tries to embed markers specific to a single user. However, there is a trade-off for the authority between the effectiveness of this process against Pirates 2.0 and the efficiency of the scheme.

Eventually, the most interesting direction is probably to provide modified versions of the common traitor tracing schemes that resist Pirates 2.0 attacks without sacrificing the efficiency of the original schemes.

References

1. AACS LA. AACS Specifications, http://www.aacsla.com/specifications
2. AACS LA. Introduction and Common Cryptographic Elements. Downloaded from, http://www.aacsla.com/specifications/specs091/AACS_Spec_Common_0.91.pdf
3. Billet, O., Phan, D.H.: Efficient Traitor Tracing from Collusion Secure Codes. In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 171–182. Springer, Heidelberg (2008)
4. Boneh, D., Franklin, M.K.: An Efficient Public Key Traitor Tracing Scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)
5. Boneh, D., Naor, M.: Traitor tracing with constant size ciphertexts (2008)
6. Boneh, D., Sahai, A., Waters, B.: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
7. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 452–465. Springer, Heidelberg (1995)
8. Chabanne, H., Phan, D.H., Pointcheval, D.: Public Traceability in Traitor Tracing Schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 542–558. Springer, Heidelberg (2005)
9. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
10. DefectiveByDesign, http://www.defectivebydesign.org/
11. Electronic Frontier Foundation, http://www.eff.org/
12. Fiat, A., Tassa, T.: Dynamic Traitor Tracing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 354–371. Springer, Heidelberg (1999)
13. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: Symposium on Foundations of Computer Science—FOCS 1984, pp. 464–479. IEEE, Los Alamitos (1984)
14. Hollmann, H.D.L., van Lint, J.H., Linnartz, J.-P.M.G., Tolhuizen, L.M.G.M.: On Codes with the Identifiable Parent Property. J. Comb. Theory, Ser. A 82(2), 121–133 (1998)
15. Kiayias, A., Pehlivanoglu, S.: Pirate Evolution: How to Make the Most of Your Traitor Keys. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 448–465. Springer, Heidelberg (2007)
16. Kiayias, A., Yung, M.: On Crafty Pirates and Foxy Tracers. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, pp. 22–39. Springer, Heidelberg (2002)
17. Kiayias, A., Yung, M.: Traitor tracing with constant transmission rate. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 450–465. Springer, Heidelberg (2002)

18. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
19. Pfitzmann, B.: Trials of Traced Traitors. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 49–64. Springer, Heidelberg (1996)
20. Phan, D.H.: Traitor tracing for stateful pirate decoders with constant ciphertext rate. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 354–365. Springer, Heidelberg (2006)
21. Phan, D.H., Safavi-Naini, R., Tonien, D.: Generic Construction of Hybrid Public Key Traitor Tracing with Full-Public-Traceability. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 264–275. Springer, Heidelberg (2006)
22. Sarkar, P., Stinson, D.R.: Frameproof and IPP codes. In: Pandu Rangan, C., Ding, C. (eds.) INDOCRYPT 2001. LNCS, vol. 2247, pp. 117–126. Springer, Heidelberg (2001)
23. Sirvent, T.: Traitor tracing scheme with constant ciphertext rate against powerful pirates. In: Augot, D., Sendrier, N., Tillich, J.-P. (eds.) Workshop on Coding and Cryptography—WCC 2007, April 2007, pp. 379–388 (2007)
24. Stop DRM Now!, `http://stopdrmnow.org/`
25. Tardos, G.: Optimal probabilistic fingerprint codes. In: ACM Symposium on Theory of Computing—STOC 2003, pp. 116–125. ACM, New York (2003)

Key Agreement from Close Secrets over Unsecured Channels

Bhavana Kanukurthi and Leonid Reyzin

Boston University Computer Science
111 Cummington St., Boston, MA 02215, USA
http://cs-people.bu.edu/bhavanak, http://www.cs.bu.edu/~reyzin

Abstract. We consider information-theoretic key agreement between two parties sharing somewhat different versions of a secret w that has relatively little entropy. Such key agreement, also known as information reconciliation and privacy amplification over unsecured channels, was shown to be theoretically feasible by Renner and Wolf (Eurocrypt 2004), although no protocol that runs in polynomial time was described. We propose a protocol that is not only polynomial-time, but actually practical, requiring only a few seconds on consumer-grade computers.

Our protocol can be seen as an interactive version of robust fuzzy extractors (Dodis et al., Crypto 2006). While robust fuzzy extractors, due to their noninteractive nature, require w to have entropy at least half its length, we have no such constraint. In fact, unlike in prior solutions, in our solution the entropy loss is essentially unrelated to the length or the entropy of w, and depends only on the security parameter.

1 Introduction

We consider the problem of information-theoretic key agreement between two parties that initially possess only correlated weak secrets. At the start of the protocol, Alice has a string w, Bob has w' that is similar, but not identical, to w, and the adversary Eve's information about w is incomplete. The goal is for Alice and Bob to agree on a shared secret key k about which Eve has no information. Security has to hold even in the case of active Eve, i.e., one who can perform the (wo)man-in-the-middle attack. It is important that the output k be as long as possible given the entropy of w (the difference between the length of k and the entropy of w is known as the entropy loss).

This setting arises, for example, when Alice and Bob have access to a (possibly) noisy channel that can be partially eavesdropped by Eve; or when a trusted server (Alice) stores the biometric of a user (Bob), and the user subsequently uses his fresh biometric reading to authenticate himself to the server; or when Alice and Bob are mobile nodes wanting to authenticate each other based on the fact that their knowledge of a location is greater than Eve's (e.g., if they are much closer to a particular location than Eve, and thus are able to observe it at higher resolution).

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 206–223, 2009.

Renner and Wolf [RW04] proposed the first protocol to solve this problem. This protocol (described in [RW04, Corollary 9]) is very general: it does not require proximity between w and w', but only requires, roughly, that information that w and w' contain about each other is more than the information that Eve has about them. However, the price for this generality is that the protocol is not practical as presented: the round complexity is quite high, and the local running time of each party is not even polynomial.

Renner and Wolf mention briefly, however [RW04, Section 2.2], that local running time can be made polynomial through the use of error-correcting techniques when w and w' "are bitstrings which differ in a certain limited number of positions" (that is, are close in the Hamming metric). Indeed, subsequently, Dodis et al. [DKRS06] used error-correcting techniques to propose a protocol that is computationally efficient not only for the Hamming metric, but also for the set difference metric. Moreover, their protocol has just a single message from Alice to Bob.

Unfortunately, the price for such high efficiency is high entropy loss: if the length of w is n and its entropy (after the error-correcting information) is m, then the protocol of Dodis et al. cannot output k longer than $m - (n - m)$. In particular, if the entropy of w is less than half its length, it achieves nothing (this is unavoidable in all single-message protocols [DKRS06, KR08a], as pointed out in [DKRS06] and shown in [DW08]).

Our Contribution. We build on the results of [RW04] and [DKRS06] by proposing a protocol that is efficient for both parties and has both lower round complexity and lower entropy loss than the protocol of [RW04]. Our analysis decouples security from the length n of w, thus offering a flexible tradeoff between security and performance. Without going into details of all the parameters, for security 2^{-L}, the length of k in our protocol is about $m - L^2/2 - O(L \log L + L \log n)$ and the number of messages exchanged between Alice and Bob is $L + \log n + 5$. More details and a more careful performance comparison are provided in Section 2.

Our protocol is more general than the work of [DKRS06] not only in the entropy requirement, but also in the kinds of differences between w and w' it can handle. Specifically, it can handle any metric that has secure sketches [DORS08] (see Section 3.2) that do not lose too much entropy (in particular, therefore, our protocol tolerates Hamming, set difference, edit distance [DORS08] and point-set difference [CL06] errors). Thus, while Renner and Wolf showed feasibility of key agreement from correlated information, and Dodis et al. showed its practicality for certain restricted settings, we demonstrate its practicality for a broad class of settings.

Implementation Results. We implemented our protocol (using Shoup's NTL [Sho01]), although we have not performed careful code optimization and did not include any improvements of Section 4.2. The protocol was tested for $L = 80$ and $n = 100,000$ on a LAN with Alice and Bob running on a 2.4Ghz Intel Pentium 4 and a WAN with Alice running on a 2.4Ghz Intel Xeon instead. The running times over a WAN and LAN were nearly the same, both less than 5 seconds. Of the total running time, approximately 1.5 seconds were spent by each party on

computation and an additional 1 second was spent in total communication costs. The improvements in Section 4.2 will reduce the running time further (although the impact of these improvements on the number of rounds and the amount of computation is easy to understand, it is difficult to say how much the actual total running times will decrease).

Other related work. Variants of this problem have been studied, under the names "information reconciliation," "privacy amplification," and "fuzzy extractors." Without providing an exhaustive overview of the literature, we note here the most closely related work. Information-theoretic security against active Eve was achieved by Maurer, Renner, and Wolf [Mau97, MW97, Wol98, MW03, RW03] in the restricted setting when $w = w'$ or when w, w', and Eve's information come from repeated independent identically distributed experiments. Boyen et al [BDK$^+$05] removed those restrictions, instead requiring that w and w' be close in some metric that has secure sketches, but achieved only computational security. One of their solutions relies on the random oracle model, and the other on computational assumptions necessary to enable password-base authenticated key agreement.

The starting point for our work is the same as for [RW04]: a protocol, also by Renner and Wolf [RW03], designed for the case of $w = w'$. We modify it for the case of $w \neq w'$ in a way that improves it even for the case of $w = w'$, and provide a more careful, concrete security analysis for it ([RW03] provides only an asymptotic analysis that works when $n \to \infty$).

2 Overview of the Result

Notation, Distributions, Entropy Let U_l denote the uniform distribution on $\{0,1\}^l$. Let X_1, X_2 be two probability distributions over some set S. Their *statistical distance* is

$$\mathbf{SD}(X_1, X_2) \stackrel{\text{def}}{=} \max_{T \subseteq S} \{\Pr[X_1 \in T] - \Pr[X_2 \in T]\} = \frac{1}{2} \sum_{s \in S} \left| \Pr_{X_1}[s] - \Pr_{X_2}[s] \right|$$

(they are said to be ε-close if $\mathbf{SD}(X_1, X_2) \leq \varepsilon$). The *min-entropy* of a random variable W is $\mathbf{H}_\infty(W) = -\log(\max_w \Pr[W = w])$ (all logarithms are base 2, unless specified otherwise). Following [DORS08], for a joint distribution (W, E), define the (average) conditional min-entropy of W given E as

$$\widetilde{\mathbf{H}}_\infty(W \mid E) = -\log(\mathop{\mathbf{E}}_{e \leftarrow E}(2^{-\mathbf{H}_\infty(W|E=e)}))$$

(here the expectation is taken over e for which $Pr[E = e]$ is nonzero). A computationally unbounded adversary who receives the value of E cannot find the correct value of W with probability greater than $2^{-\widetilde{\mathbf{H}}_\infty(W|E)}$.

Throughout this paper, for any string x, we use the notation λ_x to denote its length and h_x to denote its entropy (i.e, $\mathbf{H}_\infty(X)$).

Model We now define our goal, by modifying the noninteractive robust fuzzy extractor definition of [DKRS06]. An *Interactive Robust Fuzzy Extractor* protocol allows two parties, Alice and Bob, holding instances w, w' of correlated random variables W, W' that are guaranteed to be close but not identical, to agree on a secret key. We assume that w and w' are within distance at most η in some underlying metric space. The correctness of the protocol guarantees that when the protocol is executed in the presence of a passive adversary (one who does not interfere with messages between Alice and Bob), the parties end up agreeing on the same secret key, as long as $\mathsf{dis}(w, w') \leq \eta$.

The security of the protocol guarantees that even when the protocol is executed in the presence of an active adversary, who interferes with messages arbitrarily, if both parties accept, then they agree on a key that is uniformly random from the adversary's point of view. Moreover, if only one party accepts, then its key is still uniformly random from the adversary's point of view. (As was observed in, for example, [Wol98] and [Sho99], we cannot require that if one party rejects, then so does the other party, because an active adversary can always replace the last message with an invalid one—by that time, the sender of that message must have already accepted, while the recipient will reject.)

More formally, let $w, w' \in \{0,1\}^n$ chosen according to distributions W, W' be the secret values held by Alice and Bob respectively. Call three correlated random variables (W, W', E) (where W and W' range over some metric space $\mathcal{M}$) *suitable* if $\widetilde{\mathbf{H}}_\infty(W \mid E) \geq h_W$ and $\Pr_{(w,w') \leftarrow (W,W')}[\mathsf{dis}(w, w') \leq \eta] = 1$. Let Protocol (A, B) be executed in the presence of an active adversary Eve. Let C_a be the random variable describing A's view of the communication when (A, B) is executed in the presence of Eve.Likewise, define C_b. (We will use c_a, c_b to denote specific values of these variables.) We denote the private coins of Alice and Bob by r_a and r_b respectively. Alice's output will be denoted by $\mathsf{k_A} = \mathsf{A}(w, c_a, r_a)$, and Bob's by $\mathsf{k_B} = \mathsf{B}(w', c_b, r_b)$ (if successful, both will be of length λ_k; rejection will be denoted by a special symbol $\perp$). Let $C = C_a \cup C_b \cup E$ be Eve's view of the protocol; because Eve is computationally unbounded, we can assume she is deterministic.

Definition 1. *An interactive protocol* (A, B) *played by* Alice *and* Bob *on a communication channel fully controlled by an adversary Eve, is an* $(\mathcal{M}, h_W, \lambda_k, \eta, \delta, \epsilon)$-interactive robust fuzzy extraction protocol *if it satisfies the following properties whenever* (W, W', E) *are suitable:*

1. *Correctness. If Eve is passive, then* $\Pr[\mathsf{k_A} = \mathsf{k_B}] = 1$.
2. *Robustness. The probability that the following experiment outputs "Eve wins" is at most* δ*: sample* (w, w', e) *from* (W, W', E)*; let* c_a, c_b *be the communication upon execution of* (A, B) *with Eve*(e) *actively controlling the channel, and let* $\mathsf{A}(w, c_a, r_a) = \mathsf{k_A}, \mathsf{B}(w', c_b, r_b) = \mathsf{k_B}$*. Output "Eve wins" if* $(\mathsf{k_A} \neq \mathsf{k_B} \wedge \mathsf{k_A} \neq \perp \wedge \mathsf{k_B} \neq \perp)$.
3. *Extraction. Letting* C *denote an active Eve's view of the protocol,*

$$\mathbf{SD}\left((\mathsf{k_A}, C, E \mid \mathsf{k_A} \neq \perp), (U_{\lambda_k}, C, E)\right) \leq \epsilon \text{ and}$$

$$\mathbf{SD}\left((\mathsf{k_B}, C, E \mid \mathsf{k_B} \neq \perp), (U_{\lambda_k}, C, E)\right) \leq \epsilon\,.$$

Our Protocol. Before going into details in subsequent sections, we present here a high-level overview of our protocol. We start with an authentication sub-protocol Auth presented in [RW03] that achieves the following: using the secret w that is common to Alice and Bob, it allows Alice to send to Bob an authentic (but nonsecret) message M of length λ_M bit-by-bit in $2\lambda_M$ messages. Alice and Bob [RW03] can use this sub-protocol in order to agree on a key k as follows: they use Auth to get an extractor seed s from Alice to Bob, and then extract k from w using s.[1]

We modify this protocol by using Auth to authenticate a MAC key instead of an extractor seed. The MAC key, in turn, is used to authenticate the extractor seed s (which can be done very efficiently using simple information-theoretic MACs). This seems counterintuitive, because Auth reveals what is being authenticated, while MAC keys need to remain secret. The insight is to *use* the MAC key *before* Auth begins.[2] Our modification is beneficial for three reasons. First, MAC keys can be made shorter than extractor keys, so Auth is used on a shorter string, thus reducing the number of rounds and the entropy loss. Second, this modification allows us to use the same MAC key to authenticate not only the extractor seed s, but also the error-correction information (the so-called "secure sketch" of w [DORS08]) in the case Bob's w' is different from Alice's w. Third, because there are MACs that are secure even against (limited) key modification [DKRS06, CDF+08], we can lower the security parameters in Auth, further increasing efficiency and reducing entropy loss.

The rest of the paper is devoted to filling in the details of the above overview, including smaller improvements not discussed here, and proving the following theorem.

Theorem 1. *Given an $[n, \kappa, 2\eta+1]_2$ linear error correcting code, the protocol presented in Section 4 is an $(\mathcal{M}, h_W, \lambda_k, \eta, \delta, \epsilon)$-*interactive robust fuzzy extraction protocol*, where $\mathcal{M}$ is the Hamming space over $\{0,1\}^n$ with the following parameters: Setting security $\delta = 2^{-L}$, the protocol can extract $\lambda_k = h_W - (n-\kappa) - 2\log\frac{1}{\varepsilon} - (L^2/2 + O(L(\log n + \log L)))$ bits (assuming $n < 2^L$ and $\lambda_k - (n-\kappa) + 2\log\frac{1}{\varepsilon} > 10L$). The protocol involves an exchange of $L + \log n + 5$ messages between the two parties.*

The constant hidden by the O in the entropy loss is small, with $O(L(\log n + \log L))$ really being less than $3L\log 2L + \frac{1}{2}L\log n + 3(\log 8L)(\log 16n)$.

We obtain similar results for other metric spaces, with the only difference being that $n - \kappa$ in the entropy loss gets replaced by the entropy loss of the secure sketch for that metric space (see Section 3.2).

Comparison with Prior Work. When no error-correction is needed (i.e., $w = w'$ and $\eta = 0$), then $n - \kappa = 0$, and we get an improvement of the result of [RW03].

[1] For technical reasons, since the adversary can modify message of Auth, she may have some information about the string extracted from w; this problem is easily handled, see Section 4.

[2] This idea has been used before in several contexts; to the best of our knowledge it was first used in [Che97] in the context of secure link state routing.

The result of [RW03] sets $L = \Theta(\sqrt{n}/\log n)$ and loses $\Theta(n/\log n)$ bits of entropy. This can be seen in the description of protocol Auth in [RW03], which has $\Theta(\sqrt{n})$ rounds, each losing $\Theta(L)$ bits. Our protocol has only $\Theta(L)$ rounds, with each also losing $\Theta(L)$ bits. Thus, our result is a $\Theta(\log n)$-factor improvement in efficiency and entropy loss for the same security (moreover, the constant hidden by Θ, although difficult to compute exactly, is substantial, likely bigger than $\log n$ in real applications).

A precise comparison with [RW04], which uses [RW03] as a building block and adds error-correction, is even more complicated. Our advantage in the number of rounds remains the same, though the constant factor improves even further. To compare the entropy loss, we can fix the secure sketch code used in our protocol (which can be based on any linear error-correcting code) to the one implicitly used in [RW04]. In that case, the entropy loss due to added error-correction is asymptotically the same for our protocol and for the protocol of [RW04], though the constant in our protocol is substantially lower. On the other hand, an important advantage of our protocol is that we can choose a code that is efficiently decodable, in which case the entropy loss due to error-correction may increase, but the protocol will run in polynomial-time.

We now compare our result to the construction of [DKRS06]. The advantage of the [DKRS06] construction is that it takes only a single message and the entropy loss is linear in L rather than quadratic. The disadvantage is that it loses additional $n - h_W$ bits of entropy, which means that it is most effective when W has very high entropy. In particular, it becomes inapplicable when $h_W \leq n/2$.

3 Building Blocks

3.1 Extractors

Because in this paper Eve is always assumed to have some external information E about Alice and Bob's secrets, we need the following variant, defined in [DORS08, Definition 2], of the definition of strong extractors of [NZ96]:

Definition 2. *Let* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^l$ *be a polynomial time probabilistic function that uses* r *bits of randomness. We say that* Ext *is an* average-case (n, m, l, ε)-strong extractor *if for all pairs of random variables* (W, E) *such that* $w \in W$ *is an* n*-bit string and* $\widetilde{\mathbf{H}}_\infty(W \mid E) \geq m$*, we have* $\mathbf{SD}((\mathsf{Ext}(W; X), X, E), (U_l, X, E)) \leq \varepsilon$*, where* X *is the uniform distribution over* $\{0,1\}^r$.

We should note that some strong extractors (in particular, universal hashing [CW79, HILL99]) are already average-case extractors, and any strong extractor can be made average-case with a slight increase in input entropy [DORS08, Section 2.5].

The following (new) lemma shows that strings extracted by average-case extractors have high average min-entropy, even given the seed. The proof can be found in the full version [KR08b].

Lemma 1. *Let* Ext *be a an average-case* (n, m, l, ε)*-strong extractor. Then if* $\widetilde{\mathbf{H}}_\infty(W \mid E) \geq m$*, and* W *consists of* n*-bit strings,* $\widetilde{\mathbf{H}}_\infty(\mathsf{Ext}(W, X) \mid X, E) \geq \min\left(l, \log \frac{1}{\varepsilon}\right) - 1$.

3.2 A Variation on Secure Sketches

So far, we have presented the error-correcting information that Alice sends to Bob in the first message as a secure sketch. Actually, we need a slight variant on secure sketches, one that provides some resilience even when the sketch is modified. The requires a different definition than the definition of [DORS08], though it turns out that known constructions need to be modified only slightly to satisfy it.

Secure sketches, defined in [DORS08], provide two algorithms: "generate" (Gen) that takes an input w and produces a sketch P and "recover" (Rec) that outputs w from the sketch P and any w' sufficiently close to w. Their security guarantees that some entropy remains in w even given P. Secure sketches provide no guarantees when P has been tampered with, while we need to make sure that the output of Rec still has entropy. Thus, we need to add a weak form of robustness (i.e., resilience to active attack) to secure sketches. At the same time, we do not need a full recovery of the original w: we will be satisfied if both Gen and Rec produce some string R that preserves some of the entropy of w. In that way, our new primitive is like a fuzzy extractor, except we do not require that R be uniform, merely that it have entropy. In keeping with extractor literature terminology [CRVW02], we call the primitive a *weakly robust fuzzy conductor* because it conducts entropy from w to R and is robust against active attacks on P. Because we no longer recover the original w but rather reproduce the same R, we rename Rec into Rep.

Let $\mathcal{M}$ be a metric space with distance function dis. Suppose $(\mathsf{Gen}, \mathsf{Rep})$ are two procedures, where $\mathsf{Gen}(w)$, for $w \in \mathcal{M}$, outputs an extracted string $R \in \{0,1\}^*$ and a helper string $P \in \{0,1\}^*$, and $\mathsf{Rep}(w', P')$, for $w' \in \mathcal{M}, P' \in \{0,1\}^*$, outputs $R' \in \{0,1\}^*$.

Definition 3. *The procedures* $(\mathsf{Gen}, \mathsf{Rep})$ *are an* $(\mathcal{M}, h_W, h_R, h_{R'}, \eta)$*-weakly robust fuzzy conductor if they satisfy the following properties:*

1. *Error-Tolerance. If* $\mathsf{dis}(w, w') \leq \eta$ *and* R, P *were generated by* $(R, P) \leftarrow \mathsf{Gen}(w)$*, then* $\mathsf{Rep}(w', P) = R$.
2. *Security of* Gen. *For any suitable* (W, W', E)*, the string* R *has high entropy even for those who observe* P *and* E*: if* $(R, P) \leftarrow \mathsf{Gen}(W)$*, then* $\widetilde{\mathbf{H}}_\infty(R \mid E, P) \geq h_R$.
3. *Security of* Rep. *Even if the adversary modifies* P*, the string produced by* Rep *has high entropy: for all (adversarial) functions* A *and suitable* (W, W', E)*, if* $(R, P) \leftarrow \mathsf{Gen}(W)$*,* $P' \leftarrow A(P, E)$*, and* $R' \leftarrow \mathsf{Rep}(W', P')$*, then* $\widetilde{\mathbf{H}}_\infty(R' \mid E, P) \geq h_{R'}$.

We can build weakly robust fuzzy conductors out of any secure sketch $(\mathsf{SS}, \mathsf{Rec})$. (Secure sketches, defined in [DORS08], allow the recovery of w from a close

string w'). We use the secure sketch constructions of [DORS08] to build weakly robust fuzzy conductors for Hamming, set difference, and edit distance metrics. Namely, in Appendix A, we easily obtain

- for Hamming distance over an alphabet of size F, given an $[n, \kappa, 2t+1]$ linear error-correcting code for the alphabet, we get $h_R = h_W - (n-\kappa)\log F$, $h_{R'} = h_W - 2(n-\kappa)\log F$, and $\eta = t$.
- for set difference, with sets whose elements come from a universe of size U, we get $h_R = h_W - \eta\log(U+1)$ and $h_{R'} = h_W - 2\eta\log(U+1)$ for any η.
- for edit distance over an alphabet of size we get $h_R = h_W - \lceil \frac{n}{c} \rceil \log(n-c+1) - \alpha$, and $h_{R'} = h_W - \lceil \frac{n}{c} \rceil \log(n-c+1) - 2\alpha$, where $\alpha = (2c-1)\eta\lceil\log(F^c+1)\rceil$, for any constant c and η.

3.3 One-Time Message Authentication Codes (MACs)

One-time MACs allow information-theoretic authentication of a message using a key shared in advance.

Definition 4. *A function family* $\{\mathsf{MAC}_k : \{0,1\}^{\lambda_M} \to \{0,1\}^{\lambda_\sigma}\}$ *is a* δ*-secure one-time deterministic MAC for messages of length* λ_M *with tags of length* λ_σ *if for any* $M \in \{0,1\}^{\lambda_M}$ *and any function (adversary)* $A : \{0,1\}^{\lambda_\sigma} \to \{0,1\}^{\lambda_M} \times \{0,1\}^{\lambda_\sigma}$,

$$\Pr_k [\mathsf{MAC}_k(M') = \sigma' \wedge M' \neq M \mid (M', \sigma') = A(\mathsf{MAC}_k(M))] \leq \delta\,.$$

MAC Construction. We will use the following standard MAC technique [dB93], [Tay93], [BJKS93]. View the key k as two values, a and b, of λ_σ bits each. Split the message M into c chunks $M_0, \ldots, M_{c-1}$, each λ_σ bits long, and view these as coefficients of a polynomial $\tilde{M}(x) \in \mathbb{F}_{2^{\lambda_\sigma}}[x]$ of degree $c-1$. Then $\mathsf{MAC}_k(M) \stackrel{\text{def}}{=} a\tilde{M}(a) + b$. This is a $\lceil \lambda_M/\lambda_\sigma \rceil\, 2^{-\lambda_\sigma}$-secure message authentication code.

This construction has two properties that are particularly important to us. First, its key length is close to optimal (it is not hard to show that $\lambda_\sigma \geq \log\frac{1}{\delta}$—else, adversary could simply guess a tag; and $|k| \geq 2\log\frac{1}{\delta}$—else, there would be fewer than $\frac{1}{\delta}$ tags for M' given one of the $\frac{1}{\delta}$ tags for M). Second, it is secure even when the adversary knows something about the key, with security degrading according to the amount of information adversary knows (this kind of security was first addressed in [MW97]). Intuitively, the security of this MAC is roughly the entropy of the key minus half the key length. More formally,

Proposition 1. *Let* (K, E) *be a joint distribution. Then for all (adversarial) functions* M *with* λ_M*-bit outputs and* A,

$$\Pr_{(k,e)\leftarrow(K,E)} [\mathsf{MAC}_k(M') = \sigma' \wedge M' \neq M \mid (M', \sigma') = A(\mathsf{MAC}_k(M(e)), e)] \leq \left\lceil \frac{\lambda_M}{\lambda_\sigma} \right\rceil 2^{\lambda_\sigma - \tilde{\mathbf{H}}_\infty(K|E)}\,.$$

We defer the proof of this proposition to the full version [KR08b]. We note that its proof becomes simpler (than similar prior proofs) if we use the notion of average min-entropy. In particular, we will use following lemma [DORS08] that states that average min-entropy of a variable from the point of view of an adversary doesn't decrease by more than the number of bits (correlated with the variable) observed by the adversary.

Lemma 2. *If B has at most 2^λ possible values, then*

$$\widetilde{\mathbf{H}}_\infty(A \mid B, E) \geq \widetilde{\mathbf{H}}_\infty(A, B \mid E) - \lambda \geq \widetilde{\mathbf{H}}_\infty(A \mid E) - \lambda$$

3.4 A Modification of Renner-Wolf Interactive Authentication

The [RW03] authentication protocol allows two parties who share the same string R to authenticate a message M, even if R has very little entropy.

We generalize this protocol slightly (to use general extractors instead of the specific polynomial authentication function) and present it in Figure 1. We assume that Ext is an average-case extractor that takes seeds of length q, and outputs $L+1$-bit strings that are 2^{-L-1}-close to uniform as long as the input has sufficient entropy h (in particular, $h \geq 3L+1$ suffices if one is using universal hashing as the extractor). For our purposes, it suffices to assume that the length of M and the number of ones in it (i.e., its Hamming weight $\mathsf{wt}(M)$) are known to Bob. If $|M|$ is known but $\mathsf{wt}(M)$ is not, M can be first encoded as a balanced

Alice and Bob share a string R. Alice wishes to authentically send Bob $M = M_1 \ldots M_{\lambda_M}$ of λ_M bits. The value λ_M and the number of ones in M is known to Bob.

For i = 1 to λ_M:

1. Alice sends Bob challenge $x_i \in_r \{0,1\}^q$.

2. Bob receives x'_i, sends $b'_i = \mathsf{Ext}(R; x'_i)$, and challenge $y'_i \in_r \{0,1\}^q$

3. Alice receives b_i, y_i, verifies that $b_i = \mathsf{Ext}(R; x_i)$ and aborts if not. She sends $(1, a_i = \mathsf{Ext}(R; y_i))$ if $M_i = 1$, and $(0, \perp)$ otherwise.

4. Bob receives b'_i, a'_i aborts if $b'_i = 1$ and $a'_i \neq \mathsf{Ext}(R; y'_i)$; accepts otherwise. If $i = \lambda_M$, Bob verifies that the number of ones in the received string match the expected number of ones; aborts otherwise.

Note that step 3 and 4 of each iteration are combined with steps 1 and 2, respectively, of the next iteration.

Fig. 1. Protocol Interactive Message Authentication Auth

string (i.e., a string with the same number of zeros and ones), by encoding, for example, a 0 as 01 and a 1 as a 10. This doubles the length of M.[3]

We note that [RW03] present a technique that can be used even if $|M|$ is unknown (namely, encoding M as a string that becomes balanced only at the end), but we will not need it here.

Each round of the protocol reveals $L+1$ bits of information correlated to R if $M_i = 0$, and $2L+1$ bits of information of information correlated to R if $M_i = 1$. Hence, by Lemma 2, the adversary's uncertainty about R will be sufficient for the extractor to work until the last round as long as $\widetilde{\mathbf{H}}_\infty(R|E) \geq 3L+1+(L+1)(\lambda_M + \mathsf{wt}(M))$, and by Lemma 1 the a_i and b_i values will have entropy L from the adversary's point of view.

The intuition for the security of this protocol is that Eve cannot answer a random query x_i or y_i with probability greater than 2^{-L} because of the entropy of the answers, and hence can neither remove zero bits (because challenges to Bob keep him synchronized) nor insert one bits (because Alice is required to answer a challenge for each one). She can insert zero bits and change zeros to ones, but that is taken care of by the assumption that Bob knows λ_M and $\mathsf{wt}(M)$.

We do not formally define or prove security of this protocol, as the proof is essentially the same as in [RW03]. The probability that Eve succeeds in transmitting $M' \neq M$ to Bob and Bob does not reject (or Alice rejects and Bob accepts) is at most 2^{-L}.

We note the following security property observed in [RW04]. Consider a setting where, because of Eve's malicious interference, Bob does not have the same R as Alice does, but instead some (possibly correlated) R'. The protocol may not be complete, of course. However, it still secure, in the sense that Eve's chances of authenticating a message $M' \neq M$ are not more than when R is the same for Alice and Bob, as long as R' also has sufficient entropy ($\geq 3L+1+(L+1)(\lambda_M + \mathsf{wt}(M))$).

An additional security property (neither mentioned nor needed before) is that no information about the message M being authenticated is revealed to Eve until Bob receives the first message of the protocol. This holds with probability at least $1-2^{-L}$ even when Eve is active, because she cannot get Alice to reveal even the first bit M_1 without answering her challenge x_i, which she is unlikely to do without Bob's help.

[3] More efficient methods for encoding M as a balanced string are, of course, also possible. The length of M can be increased by less than $\log_2 |M|$ through the use of algorithms from Bos and Chaum [BC92] or Reyzin and Reyzin [RR02]. These algorithms compute a bijection between integers in $[1, \binom{n}{n/2}]$ and subsets of size $n/2$ of a set of size n. Any such subset can be viewed as a balanced string (where the i^{th} bit is set to 1 iff the i^{th} element is in the subset). Therefore, to balance a string M, it can be viewed as integer, and the subset produced by one of the above algorithms can be viewed as its balanced encoding.

4 Our Protocol

We propose the following privacy amplification protocol, in which Alice starts with w and Bob with w' such that $\mathsf{dis}(w, w') \leq \eta$. Below, MAC refers to the construction from Lemma 3.3 and Ext refers to an arbitrary average-case extractor (the choice of extractor will affect security only marginally, and will mostly affect efficiency, as we discuss below; in particular, extractors as simple as universal hashing can be used). Lengths that are currently undefined (such as of MAC keys and extractor seeds) will be set in subsequent sections in order to achieve desired security levels. In the protocol description below, extractor outputs of varying lengths and distance from uniform are needed at different stages of the protocol. We account for this variation by using two different extractors, denoted by Ext_1, Ext_2.

1. Alice generates a random MAC key k_1 and extractor seed s_1, computes $(R, P) \leftarrow \mathsf{Gen}(w)$, $\sigma_1 = \mathsf{MAC}_{k_1}(s_1, P)$, and sends $((s_1, P), \sigma_1)$ to Bob.
2. Alice initiates the Renner-Wolf message authentication protocol(Auth) for the message k_1 (suitably converted to a balanced string as indicated in Section 3.4), using R as the shared secret value.
3. Bob receives $((s_1', P'), \sigma_1')$, and computes $R' = \mathsf{Rep}(w', P')$. He responds to Alice's Auth protocol, using the string R' as the shared secret value.
4. Upon completion of Alice's side of Auth (if she has not yet rejected), Alice
   - extracts $k_2 = \mathsf{Ext}_1(R; s_1)$;
   - generates a random seed s_2;
   - sends Bob s_2 and $\sigma_2 = \mathsf{MAC}_{k_2}(s_2)$;
   - outputs her final key $\mathsf{k_A} = k_3 = \mathsf{Ext}_2(R; s_2)$.
5. Upon completion of Bob's side of the Auth with the received message k_1', and receipt of s_2', σ_2' from Alice, Bob
   - verifies the first MAC, $\mathsf{Verify}_{k_1'}((s_1', P'), \sigma_1')$ (if fail, rejects);
   - computes the key for the second MAC, $k_2' = \mathsf{Ext}_1(R'; s_1')$;
   - verifies the second MAC, $\mathsf{Verify}_{k_2'}(s_2', \sigma_2')$ (if fail, rejects);
   - outputs his final key $\mathsf{k_B} = k_3' = \mathsf{Ext}_2(R'; s_2')$.

The intuition behind the security of this protocol is in the ordering of events. First, Alice authenticates a message (s_1, P) to Bob using a MAC with a truly random key k_1 which is unknown to Eve. This ensures that Eve cannot (except with negligible probability) succeed in modifying the message while preserving the integrity of the tag. However, Bob does not know k_1, either—which means he must defer the verification of the tag σ until a later stage.

Second, after she is sure that Bob has received the message and the tag (and thus it is too late for Eve to try modifying them), Alice transmits k_1 to Bob using the Renner-Wolf authentication protocol. The protocol reveals all of k_1 to Eve, but at this point k_1 is completely useless to her, because it is too late to try to modify the message and the tag. She cannot modify k_1 (except with negligible probability), by the security of the authentication protocol. It is crucial here that the authentication protocol is secure *even if* Eve modified P (such modification

would not be detected until later), giving Alice and Bob different secrets R and R', because both R and R' have sufficiently high entropy. This enables Bob to verify the correctness of the MAC (and hence ensure that $R = R'$) at the end of the protocol.

The last steps of the protocol are a bit confusing, because instead of just outputting k_2 as the final key, Alice adds a level of indirection, using k_2 as a key to authenticate another extractor seed s_2, which is then used to extract the output key. This is similar to [RW03] and is needed because k_2, computed as $\mathsf{Ext}(w; s_1)$, is guaranteed to be close to uniform only when s_1 is a random seed independent of Eve's view. However, s_1 is revealed to Eve before Auth and an active Eve can modify the challenges within Auth (which are extractor seeds) to be correlated to s_1. By the time $\mathsf{Ext}(w; s_1)$ is computed after Auth, s_1 is not necessarily independent of Eve's view. Thus, k_2 is not necessarily suitable for the final output, although it is possible to show that it still has entropy and is therefore suitable as a MAC key. In Section 4.2 we show how to reduce the length of k_2 (and thus the entropy loss) as compared to the protocol of [RW03].

4.1 Analysis

The security parameter for our protocol is L. Through out this section, as with the rest of the paper, for any string x we use λ_x to denote the length of the x and h_x to denote its entropy (i.e, $\mathbf{H}_\infty(x)$).

Robustness We can view the protocol as consisting of two phases.

- Phase 1: Agreeing on k_2 from close secrets w, w'
- Phase 2: Using k_2 to agree final string k_3

Security of Phase 1. Suppose Eve succeeds in an active attack against Phase 1, i.e., $k_2 \neq k_2'$. There are two possibilities.

1. $k_1 = k_1'$ (Eve does not attack protocol Auth). Therefore, in order for $k_2 \neq k_2'$, either $s_1 \neq s_1'$ or $P \neq P'$. Because Bob verifies the first MAC, Eve needs to come up with a valid $((s_1', P_1'), \sigma_1')$, which she has to do when she forwards Bob his very first message. This case again gives rise to two possible options, depending on when Eve sends to Bob her guess for $((s_1', P_1'), \sigma_1')$:
   - If Eve sends it right after Alice sends $((s_1, P_1), \sigma_1)$ and her first challenge x_1 to Bob, then this is equivalent to an active attack on a MAC, because she needs to produce her "guess" for $((s_1', P_1'), \sigma_1')$, before she sees any information correlated with k_1. We denote this probability by $\Pr[\mathsf{mac}]$. For an appropriate setting of length of k_1 (namely, $2L + 2\log \lambda_{(s_1,P)}/L$, where $\lambda_{(s_1,P)}$ is the length of s_1 and P) using the MAC construction from Section 3.3, we can show that $\Pr[\mathsf{mac}] \leq 2^{-L}$.
   - If Eve sends it later, then she needs to respond to x_1. We denote this probability by $\Pr[\mathsf{random-challenge}]$. From Section 3.4, it follows that $\Pr[\mathsf{random-challenge}] \leq 2^{-L}$.

2. $k_1 \neq k_1'$: In this case, Eve has to authenticate $k_1' \neq k_1$, using Protocol Auth in order to succeed. Therefore, her chances of success in this case are bounded by her chances of succeeding in an active attack against Auth. We denote this probability by Pr[Auth]. Again, if we run Auth on the security parameter $L+1$, we can show that $\Pr[\mathsf{auth}] \leq 2^{-L}$.

Security of Phase 2. This analysis is essentially the same as in [RW03]; we improve it in the next section. The key $k_2 = \mathsf{Ext}(R, s_1)$ agreed upon by the parties at the end of Phase 1 is used in Phase 2 to authenticate a fresh extractor seed s_2 (of length λ_{s_2}) using the single message MAC scheme of Section 3.3. However, the authentication protocol of Phase 1 gives Eve the ability to query the parties and get some information about $\mathsf{Ext}(w, s_1)$, decreasing the entropy of k_2. Knowing that this decrease will be no more than the amount communicated about R during Phase 1 (which is $\Theta(L^2)$ bits), we will set the length of k_2 to be twice that plus $2L + 2\log \lambda_{s_2}/L$ to get the desired 2^{-L} security for the second MAC.

It is easy to verify by counting the entropy lost during each round that the protocol, as presented here, gives us Theorem 1 up to constant factors. (More precisely, it proves the following modification of Theorem 1: in the expression for λ_k, increase the coefficient of L^2 from 1/2 to 9, and increase the number of messages by a factor of 4.) In the next section we present a number of improvements that reduce the entropy loss by (significant) constant factors, proving Theorem 1.

4.2 Constant-Factor Improvements

In this section we propose improvements that reduce the round complexity by a factor of 4 and the entropy loss by a factor of up to 18, making this protocol considerably more practical.

Reducing the length of the extracted MAC key k_2. Note that choosing the length of k_2 as above increases the entropy loss of the protocol by almost a factor of 3. By reworking the analysis of Phase 1 using the notion of average min-entropy (similar to the analysis in proof of Proposition 1), we can show that requiring k_2 to be longer than twice the communication in Phase 1, as discussed above, is unnecessary. Using the same notation that we used in the protocol description, we let σ_2 denote the tag of the MAC. To succeed in forging it, the adversary Eve needs to successfully change σ_2 to σ_2'. In addition, in Phase 1 she is also allowed to query Alice and Bob, say, T times. Protocol Auth implicitly imposes the constraint that Eve needs to also respond to T such queries. Let us denote her queries by $(q_1, \ldots, q_T)$ and responses by $(q_1', \ldots, q_T')$. We analyze the security of phases I and II jointly by looking at the average min-entropy of $(\sigma_2', (q_1', \ldots, q_T'))$ given $(\sigma_2, (q_1, \ldots, q_T))$. It turns out to be roughly $\lambda_{k_2} - T - \lambda_{\sigma_2}$, which makes the likelihood that Eve to completes phase I and comes up with σ_2' is no more than 2^{-L} if $\lambda_{k_2} > 2L + T$.

Working Base 4. Recall that in ith round of Auth, Bob sends Alice an extractor seed sufficient to extract $L+1$ bits, and Alice responds with either nothing or the extracted string, depending on the value of the ith bit of the message being

transmitted. We improve this by encoding the message transmitted by Auth (namely, the MAC key k_1) in base 4 rather than in base 2. Bob will send Alice an extractor seed sufficient to extract $3L+1$ bits, and Alice will respond with nothing, the first $L+1$ bits, the first $2L+1$ bits, or all $3L+1$ bits depending on the ith digit of the message. This protocol works for strings that are "balanced" in base 4: i.e., messages M of length κ_M whose base-4 digits whose digits add up to $1.5\kappa_M$. It takes κ_M rounds and loses $2.5L\kappa_M$ bits of entropy, while maintaining the same security. This improves the number of rounds by a factor 2 and the entropy loss by a factor of $3/2.5 = 1.2$, because κ_M is half of the length that M would have if written in binary (the techniques used to balance a message in base 2 are also applicable in base 4, and increase the length by essentially the same ratio).

Working in Parallel. We further improve Auth by having Alice and Bob authenticate two halves of M to each other. Namely, Alice authenticates half of M to Bob at the same time as Bob authenticates half of M to Alice. Since M was initially chosen by Alice, she first has to sends half of M to Bob to he can authenticate it back to her. This has to occur in Alice's second message, because we need to make sure that M remains secret until after Bob's first message. Note that Bob's messages for authenticating his half of M can be combined with his answers to Alice's challenges. Namely, in each round, Alice will send Bob an extractor seed sufficient to extract $4L+1$ bits, and Bob will respond with the first $L+1$ bits, the first $2L+1$ bits, or the first $3L+1$ bits, or all $4L+1$ bits depending on the appropriate digit of M.

Note that the security proof goes through without adding any new challenges from Bob to Alice (i.e., Alice's responses remain $0, L+1, 2L+1$ or $3L+1$ extracted bits long).

This improvement cuts the number of rounds essentially by a factor of 2 (except for the fact that Bob ends up one round behind), and cuts the entropy loss by a factor of 5/4=1.25 (because there are no challenges from Bob to Alice, only from Alice to Bob, and now there are half as many of those).

Not converting to/from a balanced string. Because MACs work even when the key does not have full entropy, Alice can simply choose a random balanced string for the MAC key k_1 instead of choosing fully random k_1, converting it to a balanced string for Auth, and then having Bob convert it back. The security of the MAC will remain essentially 2^{-L} if k_1 is a random string of length $2L+2\log L$ bits that is balanced when viewed in base 4 (because its entropy $\mathbf{H}_\infty(k_1)$ will be at least $2L+\log L$ by bounds on the central quadrinomial coefficient).

While this by itself is not a big improvement (on Alice's side, choosing a random balanced string is about as hard as choosing a random string of length $2L$ and converting to a balanced one of length $2L+\log L$; so the savings are mainly on Bob's side), it enables the next one.

Lowering the number of extracted bits in Auth. If we lower the bit length of the extracted strings exchanged in Auth, we increase the probability that the adversary succeeds in making a few changes to k_1. However, by using a MAC

secure against related key attacks, we can actually tolerate a few such changes. Cramer et al. [CDF+08, Corollary 2], following a construction in [DKRS06], present a MAC that is essentially as efficient as the one we use, and is secure even if the adversary is allowed to change the key by exclusive-or with an adversarially chosen string. Thus, we need to make sure that Eve's changes to k_1 can be characterized as exclusive-or with a particular string.

Namely, suppose that instead of using responses of length $0, L+1, 2L+1$, or $3L+1$, Alice uses responses of length $0, \mu+1, 2\mu+1$, or $3\mu+1$, and instead of using responses of length $L+1, 2L+1, 3L+1$, or $4L+1$, Bob uses responses of length $L+1, L+\mu+1, L+2\mu+1$, or $L+3\mu+1$, for some $\mu < L$. The fact that Bob's responses are of length at least $L+1$ ensures that Eve cannot insert or delete digits from k_1 but with probability 2^{-L}. She can, however, increase a digit with probability $2^{-\mu}$. Because we require a balanced string, any decreased digit must be compensated by an increased digit; thus, if the total sum of digit changes is γ, then the probability that Eve does not get caught is $2^{-\gamma\mu/2}$.

We are working base 4, but using MACs that are based on bit-string keys. We will convert from base 4 to base 2 using the Gray code: $0 \to 00, 1 \to 01, 2 \to 11, 3 \to 10$. This will ensure that Hamming distance between the key sent by Alice and the key received by Bob is at most γ. (If we did not use the working-in-base-4 improvement, then, of course, this would not be necessary.)

The MAC of [CDF+08] is secure when the adversary chooses the modification to the key without seeing the key. Namely, for any string Δ, the probability that the adversary can forge a MAC with the key $\mathsf{k}_1 \oplus \Delta$ is low, where the probability is taken over a random k_1. In our setting, the adversary does get to see the key, because Auth does not hide k_1. However, what helps is that Δ likely has low Hamming weight, because to achieve high Hamming weight, Eve would have to successfully respond to a number of random challenges which does not hold. Therefore, the number of possible Δ values is small, which is almost as good as having a fixed Δ.

More precisely, let π be the security of the MAC for any fixed Δ, and α be the length of the MAC key. Then there are at most $\alpha^2/2$ values of Δ of Hamming weight 2, and by the union bound, the probability that will succeed with a forgery of the MAC by changing k_1 by two bits is at most $\pi\alpha^2/2$. At the same time, the probability that Eve will then be able to change k_1 by two bits is at most $2^{-\mu}$. Letting $\mu = 2\log\alpha$, we get that Eve's probability of success overall is $\pi/2$. Similarly, there are at most $\alpha^3/3!$ values of Δ of Hamming weight 3, and the overall probability of success using any such Δ is $\pi/3!$. Continuing in this manner, we get that overall probability of Eve's success through modification of k_1 is less than $\pi(1/2! + 1/3! + 1/4! + \dots) = \pi(e-2) < \pi$ (we are using here that Δ of Hamming weight 1 is impossible because the string $k_1 \oplus \Delta$ that Bob receives must be balanced).

Now to achieve MAC security $\pi = 2^{-L}$, we need to set the length of the MAC key, to be $\alpha \le 2L + 2\log(n/L+3) + 2$, and $\mu = 2\log(\alpha) = 2\log(2L + 2\log(n/L+3)+2)$. This follows from [CDF+08, Corollary 2].

Careful counting of round complexity and entropy loss, taking into account the improvements above, gives us the statement Theorem 1.

Acknowledgments

We thank Yevgeniy Dodis and Daniel Wichs for several insightful discussions. We also thank Renato Renner and Stefan Wolf for sharing with us their thoughts on the encoding of a string into a balanced string. We are grateful to MIT's Theory of Computation Group, where some of this work was performed, for its hospitality. This work was supported in part by the U.S. National Science Foundation grants CCF-0515100 and CNS-0546614.

References

[BC92] Bos, J.N.E., Chaum, D.: Provably Unforgeable Signatures. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 1–14. Springer, Heidelberg (1993)

[BDK+05] Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure Remote Authentication Using Biometric Data. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)

[BJKS93] Bierbrauer, J., Johansson, T., Kabatianskii, G., Smeets, B.: On Families of Hash Functions via Geometric Codes and Concatenation. In: Stinson [Sti93], pp. 331–342

[CDF+08] Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 471–488. Springer, Heidelberg (2008)

[Che97] Cheung, S.: An efficient message authentication scheme for link state routing. In: 13th Annual Computer Security Applications Conference, pp. 90–98 (1997)

[CL06] Chang, E.-C., Li, Q.: Hiding Secret Points Amidst Chaff. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 59–72. Springer, Heidelberg (2006)

[CRVW02] Capalbo, M.R., Reingold, O., Vadhan, S.P., Wigderson, A.: Randomness conductors and constant-degree lossless expanders. In: IEEE Conference on Computational Complexity, p. 15 (2002)

[CW79] Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences 18, 143–154 (1979)

[dB93] den Boer, B.: A Simple and Key-Economical Unconditional Authentication Scheme. Journal of Computer Security 2, 65–71 (1993)

[DKRS06] Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 232–250. Springer, Heidelberg (2006)

[DORS08] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM Journal on Computing 38(1), 97–139 (2007); arXiv:cs/0602007

[DW08] Dodis, Y., Wichs, D.: One-round authenticated key agreement from weak secrets. Technical Report 2008/503, Cryptology ePrint archive (2008), `http://eprint.iacr.org`

[HILL99] Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: Construction of pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)

[KR08a] Kanukurthi, B., Reyzin, L.: An Improved Robust Fuzzy Extractor. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 156–171. Springer, Heidelberg (2008)

[KR08b] Kanukurthi, B., Reyzin, L.: Key agreement from close secrets over unsecured channels. Technical Report 2008/494, Cryptology ePrint archive (2008), http://eprint.iacr.org

[Mau97] Maurer, U.M.: Information-theoretically secure secret-key agreement by NOT authenticated public discussion. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 209–225. Springer, Heidelberg (1997)

[MW97] Maurer, U.M., Wolf, S.: Privacy amplification secure against active adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 307–321. Springer, Heidelberg (1997)

[MW03] Maurer, U., Wolf, S.: Secret-key agreement over unauthenticated public channels — Part III: Privacy amplification. IEEE Trans. Info. Theory 49(4), 839–851 (2003)

[NZ96] Nisan, N., Zuckerman, D.: Randomness is linear in space. Journal of Computer and System Sciences 52(1), 43–53 (1996)

[RR02] Reyzin, L., Reyzin, N.: Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 144–153. Springer, Heidelberg (2002)

[RW03] Renner, R.S., Wolf, S.: Unconditional authenticity and privacy from an arbitrarily weak secret. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 78–95. Springer, Heidelberg (2003)

[RW04] Renner, R.S., Wolf, S.: The Exact Price for Unconditionally Secure Asymmetric Cryptography. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 109–125. Springer, Heidelberg (2004)

[Sho99] Shoup, V.: On formal models for secure key exchange. Technical Report RZ 3120 (#93166), IBM Zurich Research Lab (1999), http://eprint.iacr.org/1999/012

[Sho01] Shoup, V.: Ntl: A library for doing number theory, version 5.4.2 (2001), http://www.shoup.net/ntl

[Sti93] Stinson, D.R. (ed.): CRYPTO 1993. LNCS, vol. 773, pp. 22–26. Springer, Heidelberg (1994)

[Tay93] Taylor, R.: An Integrity Check Value Algorithm for Stream Ciphers. In: Stinson [Sti93], pp. 40–48

[Wol98] Wolf, S.: Strong security against active attacks in information-theoretic secret-key agreement. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 405–419. Springer, Heidelberg (1998)

A Building Weakly Robust Fuzzy Conductors

As mentioned before, weakly robust fuzzy conductors can be built trivially out of any secure sketch $(\mathsf{SS}, \mathsf{Rec})$ defined in [DORS08]. For the sake of completeness, we review the definition below.

Definition 5. *An* $(m, \tilde{m}, t)$-secure sketch *is a pair of efficient randomized procedures* $(\mathsf{SS}, \mathsf{Rec})$ *s.t.:*

1. *The sketching procedure* SS *on input* $w \in \mathcal{M}$ *returns a bit string* $s \in \{0,1\}^*$. *The recovery procedure* Rec *takes an element* $w' \in \mathcal{M}$ *and* $s \in \{0,1\}^*$.

2. Correctness: *If* $\mathsf{dis}(w, w') \leq t$ *then* $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$.
3. Security: *For any distribution* W *over* $\mathcal{M}$ *with min-entropy* m*, the (average) min-entropy of* W *conditioned on* s *does not decrease very much. Specifically, if* $\mathbf{H}_\infty(W) \geq m$ *then* $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \geq \tilde{m}$.

The quantity $m - \tilde{m}$ *is called the* entropy loss *of the secure sketch.* ♢

To build a weakly robust fuzzy conductor from a secure sketch, simply let $\mathsf{Gen}(w) = (w, \mathsf{SS}(w))$, and $R = \mathsf{Rep}(w', P') = \mathsf{Rec}(w', P')$ unless $\mathsf{Rec}(w', P')$ fails to produce a value that is within η of w' (which can happen only if $P' \neq P$), in which case let $\mathsf{Rep}(w', P') = w'$. If the sketch length is λ, then this construction is an $(\mathcal{M}, h_W, h_W - \lambda, h_W - 2\lambda, \eta)$-weakly robust fuzzy conductor. This can be seen as follows: $\widetilde{\mathbf{H}}_\infty(R|E, P) \geq h_W - \lambda$ by Lemma 2. If $\mathsf{Rec}(w', P') = R'$, then w can be recovered from R' if one knows $\mathsf{SS}(w')$ and $\mathsf{SS}(w)$, by computing $\mathsf{Rec}(R', \mathsf{SS}(w'))$ to get w' and then $\mathsf{Rec}(w', \mathsf{SS}(w))$ to get w. Hence, $\widetilde{\mathbf{H}}_\infty(R'|E, \mathsf{SS}(w), \mathsf{SS}(w')) \geq \widetilde{\mathbf{H}}_\infty(w|E, \mathsf{SS}(w), \mathsf{SS}(w')) \geq h_W - 2\lambda$, again by Lemma 2.

This produces weakly robust fuzzy conductors for Hamming distance and set difference, using deterministic secure sketches of Constructions 3, 5, and 6 of [DORS08]. In particular, for Hamming distance over an alphabet of size F, given an $[n, k, 2t + 1]$ linear error-correcting code for the alphabet, this gives $h_R = h_W - (n - k) \log F$, $h_{R'} = h_W - 2(n - k) \log F$, and $\eta = t$. For set difference, with sets whose elements come from a universe of size n, this gives $h_R = h_W - \eta \log(n + 1)$ and $h_{R'} = h_W - 2\eta \log(n + 1)$ for any η. Construction 9 of [DORS08], for edit distance, needs to modified slightly by omitting the information required to reverse the embedding, and letting R be the embedded version of w, $R = \mathsf{SH}_c(w)$; this produces a weakly robust fuzzy conductor for edit distance over alphabet of size F with $\lambda = (2c - 1)\eta\lceil \log(F^c + 1) \rceil$, $h_R = h_W - \lceil \frac{n}{c} \rceil \log(n - c + 1) - \lambda$, and $h_{R'} = h_W - \lceil \frac{n}{c} \rceil \log(n - c + 1) - 2\lambda$, for any choice of positive integer c and any η.

Order-Preserving Symmetric Encryption

Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill

Georgia Institute of Technology, Atlanta, GA, USA
{sasha,nchenette}@gatech.edu, {younho,amoneill}@cc.gatech.edu

Abstract. We initiate the cryptographic study of order-preserving symmetric encryption (OPE), a primitive suggested in the database community by Agrawal et al. (SIGMOD '04) for allowing efficient range queries on encrypted data. Interestingly, we first show that a straightforward relaxation of standard security notions for encryption such as indistinguishability against chosen-plaintext attack (IND-CPA) is unachievable by a practical OPE scheme. Instead, we propose a security notion in the spirit of pseudorandom functions (PRFs) and related primitives asking that an OPE scheme look "as-random-as-possible" subject to the order-preserving constraint. We then design an efficient OPE scheme and prove its security under our notion based on pseudorandomness of an underlying blockcipher. Our construction is based on a natural relation we uncover between a random order-preserving function and the hypergeometric probability distribution. In particular, it makes black-box use of an efficient sampling algorithm for the latter.

1 Introduction

Motivation. The concept of order-preserving symmetric encryption (OPE) was introduced in the database community by Agrawal et al. [1]. These are deterministic encryption schemes (aka. ciphers) whose encryption function preserves numerical ordering of the plaintexts. The reason for interest in such schemes is that they allow efficient range queries on encrypted data. That is, a remote untrusted database server is able to index the (sensitive) data it receives, in encrypted form, in a data structure that permits efficient range queries (asking the server to return ciphertexts in the database whose decryptions fall within a given range, say $[a, b]$). By "efficient" we mean in time logarithmic (or at least sub-linear) in the size of the database, as performing linear work on each query is prohibitively slow in practice for large databases.

In fact, OPE not only allows efficient range queries, but allows indexing and query processing to be done exactly and as efficiently as for unencrypted data, since a query just consists of the encryptions of a and b and the server can locate the desired ciphertexts in logarithmic-time via standard tree-based data structures. Indeed, subsequent to its publication, [1] has been referenced widely in the database community, and OPE has also been suggested for use in in-network aggregation on encrypted data in sensor networks [28] and as a tool for applying signal processing techniques to multimedia content protection [13]. Yet

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 224–241, 2009.

a cryptographic study of OPE in the provable-security tradition never appeared. Our work aims to begin to remedy this situation.

Related Work. Our work extends a recent line of research in the cryptographic community addressing efficient (sub-linear time) search on encrypted data, which has been addressed by [2] in the symmetric-key setting and [5,10,6] in the public-key setting. However, these works focus mainly on simple exact-match queries. Development and analysis of schemes allowing more complex query types that are used in practice (e.g. range queries) has remained open.

The work of [22] suggested enabling efficient range queries on encrypted data not by using OPE but so-called *prefix-preserving encryption* (PPE) [29,4]. Unfortunately, as discussed in [22,2], PPE schemes are subject to certain attacks in this context; particular queries can completely reveal some of the underlying plaintexts in the database. Moreover, their use necessitates specialized data structures and query formats, which practitioners would prefer to avoid.

Allowing range queries on encrypted data in the public-key setting was studied in [11,26]. While their schemes provably provide strong security, they are not efficient in our setting, requiring to scan the whole database on every query.

Finally, we clarify that [1], in addition to suggesting the OPE primitive, *does* provide a construction. However, the construction is rather ad-hoc and has certain limitations, namely its encryption algorithm must take as input all the plaintexts in the database. It is not always practical to assume that users know all these plaintexts in advance, so a stateless scheme whose encryption algorithm can process single plaintexts on the fly is preferable. Moreover, [1] does not define security nor provide any formal security analysis.

Defining security of OPE. Our first goal is to devise a rigorous definition of security that OPE schemes should satisfy. Of course, such schemes cannot satisfy all the standard notions of security, such as indistinguishability against chosen-plaintext attack (IND-CPA), as they are not only deterministic, but also leak the order-relations among the plaintexts. So, although we cannot target for the strongest security level, we want to define the best possible security under the order-preserving constraint that the target-applications require. (Such an approach was taken previously in the case of deterministic public-key encryption [5,10,6], on-line ciphers [4], and deterministic authenticated encryption [25].)

Weakening IND-CPA. One approach is to try to weaken the IND-CPA definition appropriately. Indeed, in the case of deterministic symmetric encryption this was done by [7], which formalizes a notion called *indistinguishability under distinct chosen-plaintext attack* or IND-DCPA. (The notion was subsequently applied to MACs in [3].) Since deterministic encryption leaks equality of plaintexts, they restrict the adversary in the IND-CPA experiment to make queries to its left-right-encryption-oracle of the form $(x_0^1, x_1^1), \ldots, (x_0^q, x_1^q)$ such that $x_0^1, \ldots, x_0^q$ are all distinct and $x_1^1, \ldots, x_1^q$ are all distinct. We generalize this to a notion we call *indistinguishability under ordered chosen-plaintext attack* or IND-OCPA, asking these sequences instead to satisfy the same *order relations*. (See Section 3.2.) Surprisingly, we go on to show that this plausible-looking definition is not very use-

ful for us, because it cannot be achieved by an OPE scheme unless the size of its ciphertext-space is exponential in the size of its plaintext-space.

An alternative approach. Instead of trying to further restrict the adversary in the IND-OCPA definition, we turn to an approach along the lines of pseudorandom functions (PRFs) or permutations (PRPs), requiring that no adversary can distinguish between oracle access to the encryption algorithm of the scheme or a corresponding "ideal" object. In our case the latter is a random order-preserving function with the same domain and range. Since order-preserving functions are injective, it also makes sense to aim for a stronger security notion that additionally gives the adversary oracle access to the decryption algorithm or the inverse function, respectively. We call the resulting notion POPF-CCA for *pseudorandom order-preserving function against chosen-ciphertext attack*.

Towards a construction. After having settled on the POPF-CCA notion, we would naturally like to construct an OPE scheme meeting it. Essentially, the encryption algorithm of such a scheme should behave similarly to an algorithm that samples a random order-preserving function from a specified domain and range on-the-fly (dynamically as new queries are made). But it is not immediately clear how this can be done; blockciphers, our usual tool in the symmetric-key setting, do not seem helpful in preserving plaintext order. Our construction takes a different route, borrowing some tools from probability theory. We first uncover a relation between a random order-preserving function and the hypergeometric (HG) and negative hypergeometric (NHG) probability distributions.

The connection to NHG. To gain some intuition, first observe that any order-preserving function f from $\{1,\ldots,M\}$ to $\{1,\ldots,N\}$ can be uniquely represented by a combination of M out of N ordered items (see Proposition 1). Now let us recall a probability distribution that deals with selections of such combinations. Imagine we have N balls in a bin, out of which M are black and $N-M$ are white. At each step, we draw a ball at random without replacement. Consider the random variable Y describing the total number of balls in our sample after we collect the x-th black ball. This random variable follows the so-called negative hypergeometric (NHG) distribution. Using our representation of an order-preserving function, it is not hard to show that $f(x)$ for a given point $x \in \{1,\ldots,M\}$ has a NHG distribution over a random choice of f. Assuming an efficient sampling algorithm for the NHG distribution, this gives a rough idea for a scheme, but there are still many subtleties to take care of.

Handling multiple points. First, assigning multiple plaintexts to ciphertexts independently according to the NHG distribution cannot work, because the resulting encryption function is unlikely to even be order-preserving. One could try to fix this by keeping tracking of all previously encrypted plaintexts and their ciphertexts (in both the encryption and decryption algorithms) and adjusting the parameters of the NHG sampling algorithm appropriately for each new plaintext. But we want a stateless scheme, so it cannot keep track of such previous assignments.

Eliminating the state. As a first step towards eliminating the state, we show that by assigning ciphertexts to plaintexts in a more organized fashion, the state can actually consist of a static but exponentially long random tape. The idea is that, to encrypt plaintext x, the encryption algorithm performs a binary search down to x. That is, it first assigns $\mathcal{E}nc(K, M/2)$, then $\mathcal{E}nc(K, M/4)$ if $m < M/2$ and $\mathcal{E}nc(K, 3M/4)$ otherwise, and so on, until $\mathcal{E}nc(K, x)$ is assigned. Crucially, each ciphertext assignment is made according to the output of the NHG sampling algorithm run on appropriate parameters and *coins from an associated portion of the random tape indexed by the plaintext.* (The decryption algorithm can be defined similarly.) Now, it may not be clear that the resulting scheme induces a *random* order-preserving function from the plaintext to ciphertext-space (does its distribution get skewed by the binary search?), but we prove (by strong induction on the size of the plaintext-space) that this is indeed the case.

Of course, instead of making the long random tape the secret key K for our scheme, we can make it the key for a PRF and generate portions of the tape dynamically as needed. However, coming up with a practical PRF construction to use here requires some care. For efficiency it should be blockcipher-based. Since the size of parameters to the NHG sampling algorithm as well as the number of random coins it needs varies during the binary search, and also because such a construction seems useful in general, it should be both variable input-length (VIL) and variable output-length, which we call a *length-flexible* (LF)-PRF. We propose a generic construction of an LF-PRF from a VIL-PRF and a (keyless) VOL-PRG (pseudorandom generator). Efficient blockcipher-based VIL-PRFs are known, and we suggest a highly efficient blockcipher-based VOL-PRG that is apparently folklore. POPF-CCA security of the resulting OPE scheme can then be easily proved assuming only standard security (pseudorandomness) of an underlying blockcipher.

Switching from NHG to HG. Finally, our scheme needs an efficient sampling algorithm for the NHG distribution. Unfortunately, the existence of such an algorithm seems open. It is known that NHG can be approximated by the negative binomial distribution [24], which in turn can be sampled efficiently [16,14], and that the approximation improves as M and N grow. However, quantifying the quality of approximation for fixed parameters seems difficult.

Instead, we turn to a related probability distribution, namely the hypergeometric (HG) distribution, for which a very efficient exact (not approximated) sampling algorithm is known [20,21]. In our balls-and-bin model with M black and $N - M$ white balls, the random variable X specifying the number of black balls in our sample as soon as y balls are picked follows the HG distribution. The scheme based on this distribution, which is the one described in the body of the paper, is rather more involved, but nearly as efficient: instead of $O(\log M) \cdot T_{\text{NHGD}}$ running-time it is $O(\log N) \cdot T_{\text{HGD}}$ (where $T_{\text{NHGD}}, T_{\text{HGD}}$ are the running-times of the sampling algorithms for the respective distributions), but we show that it is $O(\log M) \cdot T_{\text{HGD}}$ on average.

Discussion. It is important to realize that the "ideal" object in our POPF-CCA definition (a random order-preserving function), and correspondingly our OPE

construction meeting it, inherently leak some information about the underlying plaintexts. Characterizing this leakage is an important next step in the study of OPE but is outside the scope of our current paper. (Although we mention that our "big-jump attack" of Theorem 1 may provide some insight in this regard.)

The point is that practitioners have indicated their desire to use OPE schemes in order to achieve efficient range queries on encrypted data and are willing to live with its security limitations. In response, we provide a scheme meeting what we believe to be a "best-possible" security notion for OPE. This belief can be justified by noting that it is usually the case that a security notion for a cryptographic object is met by a "random" one (which is sometimes built directly into the definition, as in the case of PRFs and PRPs).

On a more general primitive. To allow efficient range queries on encrypted data, it is sufficient to have an order-preserving hash function family H (not necessarily invertible). The overall OPE scheme would then have secret key $(K_{\mathcal{E}nc}, K_H)$ where $K_{\mathcal{E}nc}$ is a key for a normal (randomized) encryption scheme and K_H is a key for H, and the encryption of x would be $\mathcal{E}nc(K_{\mathcal{E}nc}, x) \| H(K_H, x)$ (cf. efficiently searchable encryption (ESE) in [5]). Our security notion (in the CPA case) can also be applied to such H. In fact, there has been some work on hash functions that are order-preserving or have some related properties [23,15,18]. But none of these works are concerned with security in any sense. Since our OPE scheme is efficient and already invertible, we have not tried to build any secure order-preserving hash separately.

On the public-key setting. Finally, it is interesting to note that in a public-key setting one cannot expect OPE to provide any privacy at all. Indeed, given a ciphertext c computed under public key pk, anyone can decrypt c via a simple binary-search. In the symmetric-key setting a real-life adversary cannot simply encrypt messages itself, so such an attack is unlikely to be feasible.

2 Preliminaries

Notation and conventions. We refer to members of $\{0,1\}^*$ as strings. If x is a string then $|x|$ denotes its length in bits and if x, y are strings then $x \| y$ denotes an encoding from which x, y are uniquely recoverable. For $\ell \in \mathbb{N}$ we denote by 1^ℓ the string of ℓ "1" bits. If S is a set then $x \xleftarrow{\$} S$ denotes that x is selected uniformly at random from S. If A is a randomized algorithm and Coins is the set from where it draws its coins, then we write $A(x, y, \ldots)$ as shorthand for $R \xleftarrow{\$} \mathsf{Coins}\,;\ A(x, y, \ldots; R)$, where the latter denotes the result of running A on inputs $x, y, \ldots$ and coins R. And $a \xleftarrow{\$} A(x, y, \ldots)$ means that we assign to a the output of A run on inputs $x, y, \ldots$. For $a \in \mathbb{N}$ we denote by $[a]$ the set $\{1, \ldots, a\}$. For sets X and Y, if $f\colon X \to Y$ is a function, then we call X the domain, Y the range, and the set $\{f(x) \mid x \in X\}$ the image of the function. An adversary is an algorithm. By convention, all algorithms are required to be efficient, meaning run in (expected) polynomial-time in the length of their inputs, and their running-time includes that of any overlying experiment.

Symmetric Encryption. A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ with associated *plaintext-space* $\mathcal{D}$ and *ciphertext-space* $\mathcal{R}$ consists of three algorithms. The randomized *key generation algorithm* $\mathcal{K}$ returns a secret key K. The (possibly randomized) *encryption algorithm* $\mathcal{E}nc$ takes the secret key K, descriptions of plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$ and a plaintext m to return a ciphertext c. The deterministic *decryption algorithm* $\mathcal{D}ec$ takes the secret key K, descriptions of plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$, and a ciphertext c to return a corresponding plaintext m or a special symbol $\perp$ indicating that the ciphertext was invalid.

Note that the above syntax differs from the usual one in that we specify the plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$ explicitly; this is for convenience relative to our specific schemes. We require the usual correctness condition, namely that $\mathcal{D}ec(K, \mathcal{D}, \mathcal{R}, (\mathcal{E}nc(K, \mathcal{D}, \mathcal{R}, m)) = m$ for all K output by $\mathcal{K}$ and all $m \in \mathcal{D}$. Finally, we say that $\mathcal{SE}$ is *deterministic* if $\mathcal{E}nc$ is deterministic.

IND-CPA. Let $\mathcal{LR}(\cdot, \cdot, b)$ denote the function that on inputs m_0, m_1 returns m_b. For a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ and an adversary A and $b \in \{0, 1\}$ consider the following experiment:

$$\begin{array}{l}\textbf{Experiment } \mathbf{Exp}^{\text{ind-cpa-b}}_{\mathcal{SE}}(A) \\ \quad K \xleftarrow{\$} \mathcal{K} \\ \quad d \xleftarrow{\$} A^{\mathcal{E}nc(K, \mathcal{LR}(\cdot,\cdot,b))} \\ \quad \text{Return } d\end{array}$$

We require that each query (m_0, m_1) that A makes to its oracle satisfies $|m_0| = |m_1|$. For an adversary A, define its *ind-cpa advantage* against $\mathcal{SE}$ as

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;=\; \Pr[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A) = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A) = 1\,]\,.$$

Pseudorandom functions (PRFs). A *family of functions* is a map $F\colon Keys \times \mathcal{D} \to \{0,1\}^\ell$, where for each key $K \in Keys$ the map $F(K, \cdot)\colon \mathcal{D} \to \{0,1\}^\ell$ is a function. We refer to $F(K, \cdot)$ as an *instance* of F. For an adversary A, its *prf-advantage* against F, $\mathbf{Adv}^{\text{prf}}_F(A)$, is defined as

$$\Pr\left[\, K \xleftarrow{\$} Keys \;:\; A^{F(K,\cdot)} = 1 \,\right] - \Pr\left[\, f \xleftarrow{\$} \mathsf{Func}_{\mathcal{D},\{0,1\}^\ell} \;:\; A^{f(\cdot)} = 1 \,\right],$$

where $\mathsf{Func}_{\mathcal{D},\{0,1\}^\ell}$ denotes the set of all functions from $\mathcal{D}$ to $\{0,1\}^\ell$.

3 OPE and Its Security

3.1 Order-Preserving Encryption (OPE)

We are interested in deterministic encryption schemes that preserve numerical ordering on their plaintext-space. Let us define what we mean by this. For $A, B \subseteq \mathbb{N}$ with $|A| \le |B|$, a function $f\colon A \to B$ is *order-preserving* (aka. strictly-increasing) if for all $i, j \in A$, $f(i) > f(j)$ iff $i > j$. We say that deterministic encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ with plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$

is *order-preserving* if $\mathcal{E}nc(K,\cdot)$ is an order-preserving function from $\mathcal{D}$ to $\mathcal{R}$ for all K output by $\mathcal{K}$ (with elements of $\mathcal{D},\mathcal{R}$ interpreted as numbers, encoded as strings). Unless otherwise stated, we assume the plaintext-space is $[M]$ and the ciphertext-space is $[N]$ for some $N \geq M \in \mathbb{N}$.

3.2 Security of OPE

A FIRST TRY. Security of deterministic symmetric encryption was introduced in [7], as a notion they call *security under distinct chosen-plaintext attack (IND-DCPA)*. (It will not be important to consider CCA now.) The idea is that because deterministic encryption leaks plaintext equality, the adversary A in the IND-CPA experiment defined in Section 2 is restricted to make only *distinct* queries on either side of its oracle (as otherwise there is a trivial attack). That is, supposing A makes queries $(m_0^1, m_1^1), \ldots, (m_0^q, m_1^q)$, they require that $m_b^1, \ldots m_b^q$ are all distinct for $b \in \{0,1\}$.

Noting that any OPE scheme analogously leaks the order relations among the plaintexts, let us first try generalizing the above approach to take this into account. Namely, let us further require the above queries made by A to satisfy $m_0^i < m_0^j$ iff $m_1^i < m_1^j$ for all $1 \leq i,j \leq q$. We call such an A an *IND-OCPA adversary* for *indistinguishability under ordered chosen-plaintext attack*.

IND-OCPA IS NOT USEFUL. Defining IND-OCPA adversary seems like a plausible way to analyze security for OPE. Surprisingly, it turns out not to be too useful for us. Below, we show that IND-OCPA is unachievable by a practical order-preserving encryption scheme, in that an OPE scheme cannot be IND-OCPA unless its ciphertext-space is extremely large (exponential in the size of the plaintext-space).

Theorem 1. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ be an order-preserving encryption scheme with plaintext-space $[M]$ and ciphertext-space $[N]$ for $M, N \in \mathbb{N}$ such that $2^{k-1} \leq N < 2^k$ for some $k \in \mathbb{N}$. Then there exists an IND-OCPA adversary A against $\mathcal{SE}$ such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\geq\; 1 - \frac{2k}{M-1}\,.$$

Furthermore, A runs in time $O(\log N)$ and makes 3 oracle queries. ∎

So, k in the theorem should be almost as large as M for A's advantage to be small. The proof is in Appendix A.

DISCUSSION. The adversary in the proof of Theorem 1 uses what we call the "big-jump attack" to distinguish between ciphertexts of messages that are "very close" and "far apart." The attack shows that *any* practical OPE scheme inherently leaks more information about the plaintexts than just their ordering, namely some information about their relative distances. We return to this point later.

AN ALTERNATIVE APPROACH. Instead, we take the approach used in defining security e.g. of PRPs [17] or on-line PRPs [4], where one asks that oracle access to the function in question be indistinguishable from access to the corresponding

"ideal" random object, e.g. a random permutation or a random on-line permutation. As order-preserving functions are injective, we consider the "strong" version of such a definition where an inverse oracle is also given.

POPF-CCA. Fix an order-preserving encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ with plaintext-space $\mathcal{D}$ and ciphertext-space $\mathcal{R}$, $|\mathcal{D}| \leq |\mathcal{R}|$. For an adversary A against $\mathcal{SE}$, define its *popf-cca-advantage* (or *pseudorandom order-preserving function advantage under chosen-ciphertext attack*), $\mathbf{Adv}^{\text{popf-cca}}_{\mathcal{SE}}(A)$, against $\mathcal{SE}$ as

$$\Pr\left[K \xleftarrow{\$} \mathcal{K} \ : \ A^{\mathcal{E}nc(K,\cdot),\mathcal{D}ec(K,\cdot)} = 1\right] - \Pr\left[g \xleftarrow{\$} \mathsf{OPF}_{\mathcal{D},\mathcal{R}} \ : \ A^{g(\cdot),g^{-1}(\cdot)} = 1\right],$$

where $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$ denotes the set of all order-preserving functions from $\mathcal{D}$ to $\mathcal{R}$.

Lazy sampling. Now in order for this notion to be useful, i.e. to be able show that a scheme achieves it, we also need a way to implement A's oracles in the "ideal" experiment efficiently. In other words, we need to show how to "lazy sample" (a term from [8]) a random order-preserving function and its inverse.[1]

As shown in [8], lazy sampling of "exotic" functions with many constraints can be tricky. In the case of a random order-preserving function, it turns out that straightforward procedures—which assign a random point in the range to a queried domain point, subject to the obvious remaining constraints—do not work (that is, the resulting function is not uniformly distributed over the set of all such functions). So how can we lazy sample such a function, if it is possible at all? We address this issue next.

A caveat. Before proceeding, we note that a shortcoming of our POPF-CCA notion is it does not lead to a nice answer to the question of what information about the data is leaked by a secure OPE scheme, but only reduces this to the question of what information the "ideal object" (a random order-preserving function) leaks. Although practitioners have indicated that they are willing to live with the security limitations of OPE for its useful functionality, more precisely characterizing the latter remains an important next step before our schemes should be considered for practical deployment.

4 Lazy Sampling a Random Order-Preserving Function

In this section, we show how to lazy-sample a random order-preserving function and its inverse. This result may also be of independent interest, since the more general question of what functions can be lazy-sampled is interesting in its own right, and it may find other applications as well, e.g. to [12]. We first uncover a connection between a random order-preserving function and the hypergeometric (HG) probability distribution.

[1] For example, in the case of a random function from the set of *all* functions one can simply assign a random point from the range to each new point queried from the domain. In the case of a random permutation, the former can be chosen from the set of all previously unassigned points in the range, and lazy sampling of its inverse can be done similarly. A lazy sampling procedure for a random on-line PRP and its inverse via a tree-based characterization was given in [4].

4.1 The Hypergeometric Connection

To gain some intuition we start with the following claim.

Proposition 1. *There is bijection between the set* $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$ *containing all order-preserving functions from a domain* $\mathcal{D}$ *of size* M *to a range* $\mathcal{R}$ *of size* $N \geq M$ *and the set of all possible combinations of* M *out of* N *ordered items.*

Proof. Without loss of generality, it is enough to prove the result for domain $[M]$ and range $[N]$. Imagine a graph with its x-axis marked with integers from 1 to M and its $y = f(x)$-axis marked with integers from 1 to N. Given S, a set of M distinct integers from $[N]$, construct an order-preserving function from $[M]$ to $[N]$ by mapping each $i \in [M]$ to the ith smallest element in S. So, an M-out-of-N combination corresponds to a unique order-preserving function. On the other hand, consider an order-preserving function f from $[M]$ to $[N]$. The image of f defines a set of M distinct objects in $[N]$, so an order-preserving function corresponds to a unique M-out-of-N combination. ∎

Using the above combination-based characterization it is straightforward to justify the following equality, defined for $M, N \in \mathbb{N}$ and any $x, x+1 \in [M], y \in [N]$:

$$\Pr[\, f(x) \leq y < f(x+1) \colon f \xleftarrow{\$} \mathsf{OPF}_{[M],[N]} \,] = \frac{\binom{y}{x} \cdot \binom{N-y}{M-x}}{\binom{N}{M}} \,. \tag{1}$$

Now let us recall a particular distribution dealing with an experiment of selecting from combinations of items.

Hypergeometric distribution. Consider the following balls-and-bins model. Assume we have N balls in a bin out of which M balls are black and $N-M$ balls are white. At each step we draw a ball at random, without replacement. Consider a random variable X that describes the number of black balls chosen after a *sample size* of y balls are picked. This random variable has a hypergeometric distribution, and the probability that $X = x$ for the parameters N, M, y is

$$P_{HGD}(x; N, M, y) = \frac{\binom{y}{x} \cdot \binom{N-y}{M-x}}{\binom{N}{M}} \,.$$

Notice the equality to the right hand side of Equation (1). Intuitively, this equality means we can view constructing a random order–preserving function f from $[M]$ to $[N]$ as an experiment where we have N balls, M of which are black. Choosing balls randomly without replacement, if the y-th ball we pick is black then the least unmapped point in the domain is mapped to y under f. Of course, this experiment is too inefficient to be performed directly. But we will use the hypergeometric distribution to design procedures that efficiently and recursively lazy sample a random order-preserving function and its inverse.

4.2 The LazySample Algorithms

Here we give our algorithms **LazySample**, **LazySampleInv** that lazy sample a random order-preserving function from domain $\mathcal{D}$ to range $\mathcal{R}$, $|\mathcal{D}| \leq |\mathcal{R}|$, and its

inverse, respectively. The algorithms share and maintain joint state. We assume that both $\mathcal{D}$ and $\mathcal{R}$ are sets of consecutive integers.

Two subroutines. Our algorithms make use of two subroutines. The first, denoted HGD, takes inputs $\mathcal{D}, \mathcal{R}$, and $y \in \mathcal{R}$ to return $x \in \mathcal{D}$ such that for each $x^* \in \mathcal{D}$ we have $x = x^*$ with probability $P_{HGD}(x-d; |\mathcal{R}|, |\mathcal{D}|, y-r)$ over the coins of HGD, where $d = \min(\mathcal{D}) - 1$ and $r = \min(\mathcal{R}) - 1$. (Efficient algorithms for this exist, and we discuss them in Section 4.5.) The second, denoted GetCoins, takes inputs 1^ℓ,$\mathcal{D}$,$\mathcal{R}$, and $b\|z$, where $b \in \{0,1\}$ and $z \in \mathcal{R}$ if $b = 0$ and $z \in \mathcal{D}$ otherwise, to return $cc \in \{0,1\}^\ell$.

The algorithms. To define our algorithms, let us denote by $w \xleftarrow{cc} S$ that w is assigned a value sampled uniformly at random from set S using coins cc of length ℓ_S, where ℓ_S denotes the number of coins needed to do so. Let $\ell_1 = \ell(\mathcal{D}, \mathcal{R}, y)$ denote the number of coins needed by HGD on inputs $\mathcal{D}, \mathcal{R}, y$. Our algorithms are given in Figure 1; see below for an overview. Note that the arrays F, I, initially empty, are global and shared between the algorithms; also, for now, think of GetCoins as returning fresh random coins. We later implement it by using a PRF on the same parameters to eliminate the joint state.

Overview. To determine the image of input m, **LazySample** employs a strategy of mapping "range gaps" to "domain gaps" in a recursive, binary search manner. By "range gap" or "domain gap," we mean an imaginary barrier between two consecutive points in the range or domain, respectively. When run,

```
LazySample(D, R, m)
01 M ← |D| ; N ← |R|
02 d ← min(D) − 1 ; r ← min(R) − 1
03 y ← r + ⌊N/2⌋
04 If |D| = 1 then
05     If F[D, R, m] is undefined then
06         cc ←$ GetCoins(1^{ℓ_R}, D, R, 1‖m)
07         F[D, R, m] ←cc R
08     Return F[D, R, m]

09 If I[D, R, y] is undefined then
10     cc ←$ GetCoins(1^{ℓ_1}, D, R, 0‖y)
11     I[D, R, y] ←$ HGD(D, R, y; cc)
12 x ← d + I[D, R, y]
13 If m ≤ x then
14     D ← {d + 1, ..., x}
15     R ← {r + 1, ..., y}
16 Else
17     D ← {x + 1, ..., d + M}
18     R ← {y + 1, ..., r + N}
19 Return LazySample(D, R, m)
```

```
LazySampleInv(D, R, c)
20 M ← |D| ; N ← |R|
21 d ← min(D) − 1 ; r ← min(R) − 1
22 y ← r + ⌊N/2⌋
23 If |D| = 1 then m ← min(D)
24     If F[D, R, m] is undefined then
25         cc ←$ GetCoins(1^{ℓ_R}, D, R, 1‖m)
26         F[D, R, m] ←cc R
27     If F[D, R, m] = c then return m
28     Else return ⊥
29 If I[D, R, y] is undefined then
30     cc ←$ GetCoins(1^{ℓ_1}, D, R, 0‖y)
31     I[D, R, y] ←$ HGD(D, R, y; cc)
32 x ← d + I[D, R, y]
33 If c ≤ y then
34     D ← {d + 1, ..., x}
35     R ← {r + 1, ..., y}
36 Else
37     D ← {x + 1, ..., d + M}
38     R ← {y + 1, ..., r + N}
39 Return LazySampleInv(D, R, c)
```

Fig. 1. The **LazySample, LazySampleInv** algorithms

the algorithm first maps the middle range gap y (the gap between the middle two range points) to a domain gap. To determine the mapping, on line 11 it sets, according to the hypergeometric distribution, how many points in $\mathcal{D}$ are mapped up to range point y and stores this value in array I. (In the future the array is referenced instead of choosing this value anew.) Thus we have that $f(x) \leq y < f(x+1)$ (cf. Equation (1)), where $x = d + I[\mathcal{D}, \mathcal{R}, y]$ as computed on line 12. So, we can view the range gap between y and $y+1$ as having been mapped to the domain gap between x and $x+1$.

If the input domain point m is below (resp. above) the domain gap, the algorithm recurses on line 19 on the lower (resp. upper) half of the range and the lower (resp. upper) part of the domain, mapping further "middle" range gaps to domain gaps. This process continues until the gaps on either side of m have been mapped to by some range gaps. Finally, on line 07, the algorithm samples a range point uniformly at random from the "window" defined by the range gaps corresponding to m's neighboring domain gaps. The is result assigned to array F as the image of m under the lazy-sampled function.

4.3 Correctness

When GetCoins returns truly random coins, it is not hard to observe that **LazySample**, **LazySampleInv** are consistent and sample an order-preserving function and its inverse respectively. But we need a stronger claim; namely, that our algorithms sample a *random* order-preserving function and its inverse. We show this by arguing that any (even computationally unbounded) adversary has no advantage in distinguishing oracle access to a random order-preserving function and its inverse from that to the algorithms **LazySample**, **LazySampleInv**. The following theorem states this claim.

Theorem 2. *Suppose GetCoins returns truly random coins. Then for any (even computationally unbounded) algorithm A we have*

$$\Pr[\, A^{g(\cdot),g^{-1}(\cdot)} = 1 \,] \;=\; \Pr[\, A^{\textbf{\textit{LazySample}}(\mathcal{D},\mathcal{R},\cdot),\textbf{\textit{LazySampleInv}}(\mathcal{D},\mathcal{R},\cdot)} = 1 \,] \,,$$

where g, g^{-1} denote an order-preserving function picked at random from $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$ and its inverse, respectively. ∎

We clarify that in the theorem A's oracles for **LazySample**, **LazySampleInv** in the right-hand-side experiment share and update joint state. It is straightforward to check, via simple probability calculations, that the theorem holds for an adversary A that makes one query. The case of multiple queries is harder. The reason is that the distribution of the responses given to subsequent queries depends on which queries A has already made, and this distribution is difficult to compute directly. Instead our proof, given in the full version [9], uses strong induction in a way that parallels the recursive nature of our algorithms.

4.4 Efficiency

We characterize efficiency of our algorithms in terms of the number of recursive calls made by **LazySample** or **LazySampleInv** before termination. (The

proposition below is just stated in terms of **LazySample** for simplicity; the analogous result holds for **LazySampleInv**.)

Proposition 2. *The number of recursive calls made by* **LazySample** *is at most* $\log N + 1$ *in the worst-case and at most* $5 \log M + 12$ *on average.* ∎

Above and in similar instances later in the paper, we omit ceilings on $\log M, \log N$ for readability. The proof is in the full version [9]. Note that the algorithms make one call to HGD on each recursion, so an upper-bound on their running-times is then at most $(\log N+1) \cdot T_{\text{HGD}}$ in the worst-case and at most $(5 \log M+12) \cdot T_{\text{HGD}}$ on average, where T_{HGD} denotes the running-time of HGD on inputs of size at most $\log N$. However, this does not take into account the fact that the size of these inputs decrease on each recursion. Thus, better bounds may be obtained by analyzing the running-time of a specific realization of HGD.

4.5 Realizing HGD

An efficient implementation of sampling algorithm HGD was designed by Kachitvichyanukul and Schmeiser [20]. Their algorithm is exact; it is not an approximation by a related distribution. It is implemented in Wolfram Mathematica and other libraries, and is fast even for large parameters. However, on small parameters the algorithms of [27] perform better. Since the parameter size to HGD in our **LazySample** algorithms shrinks across the recursive calls from large to small, it could be advantageous to switch algorithms at some threshold. We refer the reader to [27,20,21,14] for more details.

We comment that the algorithms of [20] are technically only "exact" when the underlying floating-point operations can be performed to infinite precision. In practice, one has to be careful of truncation error. For simplicity, Theorem 2 did not take this into account, as in theory the error can be made arbitrarily small by increasing the precision of floating-point operations (independently of M, N). But we make this point explicit in Theorem 3 that analyzes security of our actual scheme.

5 Our OPE Scheme and Its Analysis

Algorithms **LazySample**, **LazySampleInv** cannot be directly converted into encryption and decryption procedures because they share and update a joint state, namely arrays F and I, which store the outputs of the randomized algorithm HGD. For our actual scheme, we can eliminate this shared state by implementing the subroutine GetCoins, which produces coins for HGD, as a PRF and (re-)constructing entries of F and I on-the-fly as needed. However, coming up with a practical yet provably secure construction requires some care. Below we give the details of our PRF implementation for this purpose, which we call TapeGen.

5.1 The TapeGen PRF

Length-Flexible PRFs. In practice, it is desirable that TapeGen be both variable input-length (VIL)- and variable output-length (VOL)-PRF,[2] a primitive we call a *length-flexible* (LF)-PRF. (In particular, the number of coins used by HGD can be beyond one block of an underlying blockcipher in length, ruling out the use of most practical pseudorandom VIL-MACs.) That is, LF-PRF TapeGen with key-space *Keys* takes as input a key $K \in \mathit{Keys}$, an output length 1^ℓ, and $x \in \{0,1\}^*$ to return $y \in \{0,1\}^\ell$. Define the following oracle R taking inputs 1^ℓ and $x \in \{0,1\}^*$ to return $y \in \{0,1\}^\ell$, which maintains as state an array D:

$$\begin{array}{l}\textbf{Oracle } R(1^\ell, x)\\ \quad \text{If } |D[x]| < \ell \text{ then}\\ \quad\quad r \xleftarrow{\$} \{0,1\}^{\ell - |D[x]|}\\ \quad\quad D[x] \leftarrow D[x] \| r\\ \quad \text{Return } D[x]_1 \ldots D[x]_\ell\end{array}$$

Above and in what follows, s_i denotes the i-th bit of a string s, and we require everywhere that $\ell < \ell_{\max}$ for an associated maximum output length $\ell_{\max}$. For an adversary A, define its *lf-prf-advantage* against TapeGen as

$$\mathbf{Adv}^{\text{lf-prf}}_{\mathsf{TapeGen}}(A) \;=\; \Pr[\, A^{\mathsf{TapeGen}(K,\cdot,\cdot)} = 1 \,] - \Pr[\, A^{R(\cdot,\cdot)} = 1 \,]\,,$$

where the left probability is over the random choice of $K \in \mathit{Keys}$. Most practical VIL-MACs (message authentication codes) are PRFs and are therefore VIL-PRFs, but the VOL-PRF requirement does not seem to have been addressed previously. To achieve it we suggest using a VOL-PRG (pseudorandom generator) as well. Let us define the latter.

Variable-output-length PRGs. Let G be an algorithm that on input a seed $s \in \{0,1\}^k$ and an output length 1^ℓ returns $y \in \{0,1\}^\ell$. Let $\mathcal{O}_G$ be the oracle that on input 1^ℓ chooses a random seed $s \in \{0,1\}^k$ and returns $G(s,\ell)$, and let S be the oracle that on input 1^ℓ returns a random string $r \in \{0,1\}^\ell$. For an adversary A, define its *vol-prg-advantage* against G as

$$\mathbf{Adv}^{\text{vol-prg}}_{G}(A) \;=\; \Pr[\, A^{\mathcal{O}_G(\cdot)} = 1 \,] - \Pr[\, A^{S(\cdot)} = 1 \,]\,.$$

As before, we require above that $\ell < \ell_{\max}$ for an associated maximum output length $\ell_{\max}$. Call G *consistent* if $\Pr[\, G(s,\ell') = G(s,\ell)_1 \ldots G(s,\ell)_{\ell'} \,] = 1$ for all $\ell' < \ell$, with the probability over the choice of a random seed $s \in \{0,1\}^k$. Most PRGs are consistent due to their "iterated" structure.

Our LF-PRF construction. We propose a general construction of an LF-PRF that composes a VIL-PRF with a consistent VOL-PRG, namely using the output of the former as the seed for the latter. Formally, let F be a VIL-PRF and G be a consistent VOL-PRG, and define the associated pseudorandom tape

[2] That is, a VIL-PRF takes inputs of varying lengths. A VOL-PRF produces outputs of varying lengths specified by an additional input parameter.

generation function TapeGen which on inputs $K, 1^\ell, x$ returns $G(1^\ell, F(K,x))$. The following says that TapeGen is indeed an LF-PRF if F is a VIL-PRF and G is a VOL-PRG.

Proposition 3. *Let A be an adversary against TapeGen that makes at most q queries to its oracle of total input length ℓ_{in} and total output length ℓ_{out}. Then there exists an adversary B_1 against F and an adversary B_2 against G such that*

$$\mathbf{Adv}^{\text{lf-prf}}_{\mathsf{TapeGen}}(A) \;\le\; 2 \cdot (\mathbf{Adv}^{\text{prf}}_{F}(B_1) + \mathbf{Adv}^{\text{vol-prg}}_{G}(B_2)) \;.$$

Adversaries B_1, B_2 make at most q queries of total input length ℓ_{in} or total output length ℓ_{out} to their respective oracles and run in the time of A. ∎

The proof is in [9]. Concretely, we suggest the following blockcipher-based consistent VOL-PRG for G. Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Define the associated VOL-PRG $G[E]$ with seed-length k and maximum output length $n \cdot 2^n$, where G[E] on input $s \in \{0,1\}^k$ and 1^ℓ outputs the first ℓ bits of the sequence $E(s,\langle 1\rangle)\|E(s,\langle 2\rangle)\| \ldots$ (Here $\langle i\rangle$ denotes the n-bit binary encoding of $i \in \mathbb{N}$.) The following says that $G[E]$ is a consistent VOL-PRG if E is a PRF.

Proposition 4. *Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let A be an adversary against $G[E]$ making at most q oracle queries whose responses total at most $p \cdot n$ bits. Then there is an adversary B against E such that*

$$\mathbf{Adv}^{\text{vol-prg}}_{G[E]}(A) \;\le\; 2q \cdot \mathbf{Adv}^{\text{prf}}_{E}(B) \;.$$

Adversary B makes at most p queries to its oracle and runs in the time of A. Furthermore, $G[E]$ is consistent. ∎

The proof is in [9]. Now, to instantiate the VIL-PRF F in the TapeGen construction, we suggest OMAC (aka. CMAC) [19], which is also blockcipher-based and introduces no additional assumption. Then the secret-key for TapeGen consists only of that for OMAC, which in turn consists of just one key for the underlying blockcipher (e.g. AES).

5.2 Our OPE Scheme and Its Analysis

The scheme. Let TapeGen be as above, with key-space *Keys*. Our associated order-preserving encryption scheme $\mathcal{OPE}[\mathsf{TapeGen}] = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ is defined as follows. The plaintext and ciphertext-spaces are sets of consecutive integers $\mathcal{D}, \mathcal{R}$, respectively. Algorithm $\mathcal{K}$ returns a random $K \in \mathit{Keys}$. Algorithms $\mathcal{E}nc, \mathcal{D}ec$ are the same as **LazySample**, **LazySampleInv**, respectively, except that HGD is implemented by the algorithm of [20] and GetCoins by TapeGen (so there is no need to store the elements of F and I). That is, whenever an element $I[\mathcal{D}, \mathcal{R}, y]$ is needed, it is instead computed as the output of $\text{HGD}(\mathcal{D}, \mathcal{R}, y)$ on coins $\mathsf{TapeGen}(K, 1^{\ell_1}, (\mathcal{D}, \mathcal{R}, 0\|y))$, where as before $\ell_1 = \ell(\mathcal{D}, \mathcal{R}, y)$ is the number of coins needed by HGD on inputs $\mathcal{D}, \mathcal{R}, y$, and analogously an element

$F[\mathcal{D}, \mathcal{R}, m]$ is computed by sampling a uniformly random element of $\mathcal{R}$ using coins $\mathsf{TapeGen}(K, 1^{\ell_{\mathcal{R}}}, (\mathcal{D}, \mathcal{R}, 1\|m))$. (The length parameter to $\mathsf{TapeGen}$ is just for convenience; one can always generate more output bits on-the-fly by invoking $\mathsf{TapeGen}$ again on a longer such parameter. In fact, our implementation of $\mathsf{TapeGen}$ can simply pick up where it left off instead of starting over.) The exact code is given in the full paper [9].

Security. The following theorem characterizes security of our OPE scheme, saying that it is POPF-CCA secure if $\mathsf{TapeGen}$ is a LF-PRF. Applying Proposition 4, this is reduced to pseudorandomness of an underlying blockcipher.

Theorem 3. *Let $\mathcal{OPE}[\mathsf{TapeGen}]$ be the OPE scheme defined above with plaintext-space of size M and ciphertext-space of size N. Then for any adversary A against $\mathcal{OPE}[\mathsf{TapeGen}]$ making at most q queries to its oracles combined, there is an adversary B against $\mathsf{TapeGen}$ such that*

$$\mathbf{Adv}^{\text{popf-cca}}_{\mathcal{OPE}[\mathsf{TapeGen}]}(A) \;\le\; 2 \cdot (\mathbf{Adv}^{\text{prf}}_{\mathsf{TapeGen}}(B) \;+\; \lambda)\;.$$

Adversary B makes at most $q_1 = q \cdot (\log N + 1)$ queries of size at most $5 \log N + 1$ to its oracle, whose responses total $q_1 \cdot \lambda'$ bits on average, and its running-time is that of A. Above, λ, λ' are constants depending only on HGD and the precision of the underlying floating-point computations (not on M, N). ∎

The proof is in [9]. Above, λ represents an "error term" due to the fact that the "exact" hypergeometric sampling algorithm of [20] technically requires infinite floating-point precision, which is not possible in the real world. One way to bound λ would be to bound the probability that an adversary can distinguish the used HGD sampling algorithm from the ideal (infinite precision) one.

Efficiency. The efficiency of our scheme follows from our previous analyses. Using the suggested implementation of $\mathsf{TapeGen}$ in Subsection 5.1, encryption and decryption require the time for at most $\log N + 1$ invocations of HGD on inputs of size at most $\log N$ plus at most $(5 \log M + 12) \cdot (5 \log N + \lambda' + 1)/128$ invocations of AES on average for λ' in the theorem. See [9] for the details.

5.3 On Choosing N

One way to choose the size of the ciphertext-space N for our scheme is just to ensure the number of functions $[M]$ to $[N]$ is very large, say more than 2^{80}. (We assume that the size of the plaintext-space M is given.) The number of such functions, which is given by $\binom{N}{M}$, is maximized when $M = N/2$. And, since $(N/M)^M \le \binom{N}{M}$, it is greater than 2^{80} as long as $M = N/2 > 80$. However, once we have a greater understanding of what information about the data is leaked by a random order-preserving function (the "ideal object" in our POPF-CCA definition), more sophisticated criteria might be used to select N. In fact, it would also be possible to view our scheme more as a "tool" like a blockcipher rather than a full-fledged encryption scheme itself, and to try to use it to design an OPE scheme with better security in some cases. We leave these as interesting and important directions for future work.

6 On Using the Negative Hypergeometric Distribution

In the balls-and-bins model described in Section 4.1 with M black and $N - M$ white balls in the bin, consider the random variable Y describing the total number of balls in our sample after we pick the x-th black ball. This random variable follows the *negative* hypergeometric (NHG) distribution. As we discussed in the Introduction, use of the NHG distribution instead of the HG one permits slightly simpler and more efficient lazy sampling algorithms and corresponding OPE scheme. For completeness, we specify them in the full version [9]. The problem is that they require an efficient NHG sampling algorithm, and the existence of such an algorithm is apparently open. What is known is that the NHG distribution can be approximated by the negative binomial distribution [24], the latter can be sampled efficiently [16,14], and the approximation improves as M and N grow. However, quantifying the quality of the approximation for fixed parameters seems difficult. If future work either develops an efficient exact sampling algorithm for the NHG distribution or shows that the approximation by the negative binomial distribution is sufficiently close, then our NHG-based OPE scheme could be a good alternative to the HG-based one.

Acknowledgements

We thank the anonymous reviewers of Eurocrypt 2009 for helpful comments. Alexandra Boldyreva and Adam O'Neill are supported in part by Alexandra's NSF CAREER award 0545659 and NSF Cyber Trust award 0831184. Younho Lee was supported in part by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF:2007-357-D00243). Also, he is supported by Professor Mustaque Ahamad through the funding provided by IBM ISS and AT&T.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order-preserving encryption for numeric data. In: SIGMOD 2004, pp. 563–574. ACM, New York (2004)
2. Amanatidis, G., Boldyreva, A., O'Neill, A.: Provably-secure schemes for basic query support in outsourced databases. In: DBSec 2007, pp. 14–30. Springer, Heidelberg (2007)
3. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
4. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online ciphers and the hash-CBC construction. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 292–309. Springer, Heidelberg (2001)
5. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
6. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: Definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)

7. Bellare, M., Kohno, T., Namprempre, C.: Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In: CCS 2002, pp. 1–11. ACM Press, New York (2002)
8. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
9. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption (2009), www.cc.gatech.edu/~aboldyre/publications.html
10. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
11. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
12. Cem Say, A.C., Kutsi Nircan, A.: Random generation of monotonic functions for Monte Carlo solution of qualitative differential equations. Automatica 41(5), 739–754 (2005)
13. Erkin, Z., Piva, A., Katzenbeisser, S., Lagendijk, R.L., Shokrollahi, J., Neven, G., Barni, M.: Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing. EURASIP Journal on Information Security (2007) (Article ID 78943)
14. Fishman, G.S.: Discrete-event simulation: modeling, programming, and analysis. Springer, Heidelberg (2001)
15. Fox, E.A., Chen, Q.F., Daoud, A.M., Heath, L.S.: Order-preserving minimal perfect hash functions and information retrieval. ACM Transactions on Information Systems 9(3), 281–308 (1991)
16. Gentle, J.E.: Random Number Generation and Monte Carlo Methods. Springer, Heidelberg (2003)
17. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM 33(4), 792–807 (1986)
18. Indyk, P., Motwani, R., Raghavan, P., Vempala, S.: Locality-preserving hashing in multidimensional spaces. In: STOC 1997, pp. 618–625. ACM Press, New York (1997)
19. Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
20. Kachitvichyanukul, V., Schmeiser, B.W.: Computer generation of hypergeometric random variates. Journal of Statistical Computation and Simulation 22(2), 127–145 (1985)
21. Kachitvichyanukul, V., Schmeiser, B.W.: Algorithm 668: H2PEC: sampling from the hypergeometric distribution. ACM Transactions on Mathematical Software 14(4), 397–398 (1988)
22. Li, J., Omiecinski, E.: Efficiency and security trade-off in supporting range queries on encrypted databases. In: DBSec 2005, pp. 69–83. Springer, Heidelberg (2005)
23. Linial, N., Sasson, O.: Non-expansive hashing. In: STOC 1996, pp. 509–518. ACM Press, New York (1996)
24. López-Blázquez, F., Salamanca Miño, B.: Exact and approximated relations between negative hypergeometric and negative binomial probabilities. Communications in Statistics. Theory and Methods 30(5), 957–967 (2001)
25. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)

26. Shi, E., Bethencourt, J., Chan, T.-H.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: Symposium on Security and Privacy 2007, pp. 350–364. IEEE, Los Alamitos (2007)
27. Walker, A.J.: An efficient method for generating discrete random variables with general distributions. ACM Transactions on Mathematical Software 3, 253–256 (1977)
28. Westhoff, D., Girao, J., Acharya, M.: Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. IEEE Transactions on Mobile Computing 5(10), 1417–1431 (2006)
29. Xu, J., Fan, J., Ammar, M.H., Moon, S.B.: Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In: ICNP 2002, pp. 280–289. IEEE, Los Alamitos (2002)

A Proof of Theorem 1

We introduce the following concept for the proof. For an order-preserving function $f\colon [M] \to [N]$ call $i \in \{3, \ldots, M-1\}$ a *big jump* of f if the f-distance to the next point is as big as the sum of all the previous, i.e. $f(i+1)-f(i) \geq f(i)-f(1)$. Similarly we call $i \in \{2, \ldots M-2\}$ a *big reverse-jump* of f if $f(i) - f(i-1) \geq f(M) - f(i)$. The proof uses the following simple combinatorial lemma.

Lemma 1. *Let $f\colon [M] \to [N]$ be an order-preserving function and suppose that f has k big jumps (respectively big reverse-jumps). Then $N \geq 2^k$.* ∎

For completeness, we prove the lemma in the full version [9]. We now proceed to prove the theorem.

Proof. (of Theorem 1) Consider the following ind-ocpa adversary A against $\mathcal{SE}$:

Adversary $A^{\mathcal{E}nc(K,\mathcal{LR}(\cdot,\cdot,b))}$
$m \xleftarrow{\$} \{1, \ldots, M-1\}$
$c_1 \leftarrow \mathcal{E}nc(K, \mathcal{LR}(1, m, b))$
$c_2 \leftarrow \mathcal{E}nc(K, \mathcal{LR}(m, m+1, b))$
$c_2 \leftarrow \mathcal{E}nc(K, \mathcal{LR}(m+1, M, b))$
Return 1 if $(c_3 - c_2) > (c_2 - c_1)$
Else return 0

First we claim that

$$\Pr[\,\mathbf{Exp}^{\text{ind-ocpa-1}}_{\mathcal{SE}}(A) = 1\,] \;\geq\; \frac{(M-1)-k}{M-1} \;=\; 1 - \frac{k}{M-1}\,.$$

The reason is that m is picked independently at random and if $b = 1$ then A outputs 1 just when $m+1$ is not a big reverse-jump of $\mathcal{E}nc(K, \cdot)$, and since $N \leq 2^k$ we know that $\mathcal{E}nc(K, \cdot)$ has at most k big reverse-jumps by Lemma 1. Similarly,

$$\Pr[\,\mathbf{Exp}^{\text{ind-ocpa-0}}_{\mathcal{SE}}(A) = 1\,] \;\leq\; \frac{k}{M-1}$$

because if $b = 0$ then A outputs 1 just when m is a big jump of $\mathcal{E}nc(K, \cdot)$, and since $N \leq 2^k$ we know that $\mathcal{E}nc(K, \cdot)$ has at most k big jumps by Lemma 1. Subtracting yields the theorem. Note that A only needs to pick a random element of $[M]$ and do basic operations on elements of $[N]$, which is $O(\log N)$ as claimed. ∎

A Double-Piped Mode of Operation for MACs, PRFs and PROs: Security beyond the Birthday Barrier

Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation, Japan
yasuda.kan@lab.ntt.co.jp

Abstract. We revisit the double-pipe construction introduced by Lucks at Asiacrypt 2005. Lucks originally studied the construction for iterated hash functions and showed that the approach is effective in improving security against various types of collision and (second-)preimage attacks. Instead, in this paper we apply the construction to the secret-key setting, where the underlying FIL (fixed-input-length) compression function is equipped with a dedicated key input. We make some adjustments to Lucks' original design so that now the new mode works with a single key and operates as a multi-property-preserving domain extension of MACs (message authentication codes), PRFs (pseudo-random functions) and PROs (pseudo-random oracles). Though more than twice as slow as the Merkle-Damgård construction, the double-piped mode enjoys security strengthened beyond the birthday bound, most notably, high MAC security. More specifically, when iterating an FIL-MAC whose output size is n-bit, the new double-piped mode yields an AIL-(arbitrary-input-length-)MAC with security up to $O\left(2^{5n/6}\right)$ query complexity. This bound contrasts sharply with the birthday bound of $O\left(2^{n/2}\right)$, which has been the best MAC security accomplished by earlier constructions.

Keywords: domain extension, unpredictability, unforgeability, message authentication code, MAC, birthday bound.

1 Introduction

Many of the symmetric-key cryptographic schemes are usually realized by iterating a smaller, fixed-input-length (FIL) primitive. Examples include hash functions and message authentication codes (MACs), which are often built of a compression function or of a block cipher. The underlying compression function or block cipher has also a fixed output length, say n-bit. The size n corresponds to a security parameter in more ways than one, because it defines not only the final output size of the scheme but also the size of intermediate values in the iteration. For example, in the case of the Merkle-Damgård (MD) construction [18,9] or Cipher-Block-Chaining (CBC) mode of operation [22,15], the size of the intermediate values is exactly equal to n.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 242–259, 2009.

The "small" size n of intermediate values leads to various types of attacks and to limited security of the overlying scheme. For instance, the Merkle-Damgård hash functions are known to be vulnerable to multi-collision attacks [14] and to long-message second-preimage attacks [16]. These attacks exploit the fact that internal collisions can be found with $O(2^{n/2})$ work. For iterated MACs, the internal collisions immediately yield forgery [25,26], which confines the security of the overlying MAC scheme to $O(2^{n/2})$ query complexity. This security bound is often called the birthday "barrier."

Lucks' Double-Pipe Construction. A natural approach to precluding the above-mentioned attacks is to increase the size of intermediate values and to make it larger than the final output size n. This idea was embodied by the double-pipe hash [17] proposed by Lucks. In the double-pipe hash, the size of intermediate values was doubled to $2n$ bits by invoking two applications of the compression function per message block. The double-pipe design is more than twice as slow as the MD construction, but Lucks showed that the double-pipe hashing successfully rules out various birthday-type attacks.

In this paper we apply the double-pipe construction to the secret-key setting and analyze the security of the scheme as a MAC. We also consider stronger security notions of pseudo-random functions (PRFs) and pseudo-random oracles (PROs).[1] Of particular interest is the case of being merely a secure MAC, because there has been no known construction of MAC-Pr (preserving) domain extension with security beyond the birthday barrier. So we raise the following question:

> ***Q. Can we prove that a double-piped mode provides an AIL-(arbitrary-input-length-)MAC secure beyond the birthday barrier based on the sole assumption that the underlying compression function is a secure FIL-MAC?***

Our Results. The answer to the above question turns out to be positive. We work in the dedicated-key setting [4], where the underlying compression function is equipped with a dedicated key input like a block cipher. We then make two slight modifications to Lucks' original design and show that the new double-piped mode operates as a MAC-Pr, PRF-Pr and PRO-Pr domain extension with each type of security ensured beyond the birthday bound.

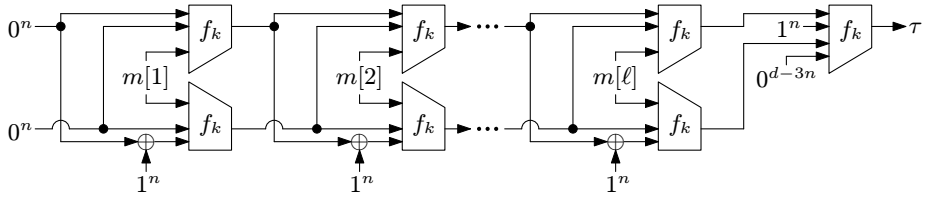


Fig. 1. Our double-piped mode of operation

[1] We use the term "PRO" [3] interchangeably with "indifferentiability" [19,5].

Figure 1 describes our altered double-piped mode of operation iterating a compression function $f_k : \{0,1\}^d \to \{0,1\}^n$ ($d \geq 3n$) with a secret key k (A formal definition of the scheme will be given in Sect. 4). The mode takes as its input a message $M = m[1] \parallel m[2] \parallel \cdots \parallel m[\ell]$ (being properly padded) with each block of $d - 2n$ bits and outputs the final value $\tau \in \{0,1\}^n$.

The mode basically follows Luck's original design, except that we make the following two minor changes:

- In the original, the chaining variable to the second application of the compression function was "tweaked" by swapping two hash values. Instead, we tweak the chaining variable by XOR-ing (rather than swapping) one of the two output values with a constant 1^n.
- In the original, the final iteration handled the last message block and worked just like intermediate iterations, except to omit the second application of the compression function. Instead, we arrange a special finalization step, which handles no message block and takes the last chaining variable "shifted" by 1^n.

The former has been already used by [6] in a more generalized form. The change is technical, and we adopt it only for simplicity of the proof. The latter technique was employed in the CS construction [20]. Unlike the first one, this change is essential to the security of our scheme. We remark that in principle the two changes do not affect Luck's original analysis, and the scheme (without the secret key) remains secure as a hash function.

The new double-piped mode of operation is highly secure; we obtain the following security results:

- **MAC-Pr.** This is the main result. We show that the scheme yields a MAC-Pr domain extension providing security beyond the birthday barrier. Our security proof involves new techniques of transforming collisions into a forgery. Using these techniques we are able to prove MAC security up to $O\big(2^{5n/6}\big)$ query complexity, improving on the previous best security of the birthday bound $O\big(2^{n/2}\big)$. We also provide a discussion on the gap between our bound $O\big(2^{5n/6}\big)$ and the full security $O\big(2^n\big)$.
- **PRF-Pr.** Our result for PRF is a straightforward corollary of that for PRO. This is due to the fact that we are working in the dedicated-key setting with the key being secret. In such a scenario, the notion of indistinguishability (*i.e.*, PRF) becomes strictly weaker than that of indifferentiability (*i.e.*, PRO). The new mode gives the full $O(2^n)$ security of PRF.
- **PRO-Pr.** We prove that the double-piped mode is indifferentiable beyond the birthday bound; the mode is secure up to $O(2^n)$ query complexity. This bound has been already realized by earlier constructions, but our scheme has slight performance advantages over them (see Sect. 2).

Organization. Section 2 reviews previous constructions of domain extensions for MACs, PRFs and PROs. Section 3 provides basic notation, adversary models, and security notions used in the paper. In Sect. 4 we give a formal definition of our double-piped mode of operation. Section 5 is devoted to the security proofs

Table 1. Comparing the MAC/PRF/PRO security of various domain extensions

	MAC	PRF	PRO	
Feistel network	$2^{n/6}$	2^n	$2^{n/16}$	[10,23,8]
Enciphered CBC	$2^{n/4}$	$2^{n/2}$	$2^{n/4}$	[11]
NI, CS, ESh, MDP	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$	[1,20,5,4,12]
Proposed double-pipe	$\mathbf{2^{5n/6}}$	$\mathbf{2^n}$	$\mathbf{2^n}$	—
Benes, multilane-NMAC, one-pass	—	2^n	—	[24,27,28]
Prefix-free, chopMD, Maurer-Tessaro	—	2^n	2^n	[6,7,21]

of our MAC-Pr result. In Sect. 6 and 7 we present the security results for PRF-Pr and PRO-Pr, respectively.

2 Related Work

In this section we go through prior constructions of MAC-Pr, PRF-Pr and PRO-Pr domain extensions. For MACs, previous constructions provide security only up to the birthday bound. For PRFs and PROs, we mention only those constructions which have security above the birthday bound. See Table 1 for summary.[2]

MAC-Pr Constructions. The study of MAC-Pr domain extension was initiated by An and Bellare [1] who presented the NI construction. The NI construction was built on the MD iteration in the dedicated-key setting. The subsequent work, including CS [20], ESh [4] and MDP [12], is all based on the MD construction. All of these constructions have the birthday-bound security $O\big(2^{n/2}\big)$. Dodis and Puniya [10] showed that the Feistel network, when iterated sufficiently many times, yields a secure, MAC domain extension with a relatively low bound of $O\big(2^{n/6}\big)$ query complexity. A year later Dodis *et al.* [11] proposed the enciphered CBC mode, which iterates block ciphers (or "length-preserving MACs") and provides an AIL-MAC with the better security of $O\big(2^{n/4}\big)$.

PRF-Pr Constructions. The problem of PRF domain extension has been extensively studied. Patarin analyzed the Feistel network [23] and the Benes network [24], and these constructions were shown to give $O(2^n)$ security. For PRF, there exist constructions which are more efficient than the double-pipe design, such as multilane-NMAC [27] and the one-pass construction in [28].

PRO-Pr Constructions. Chang *et al.* [6] proved that a double-pipe MD construction with prefix-free padding is indifferentiable beyond the birthday bound. Chang and Nandi [7] also proved that the chopMD construction based on a $2n$-bit compression function is indifferentiable beyond the birthday bound. These constructions are similar to our double-piped mode of operation, but our construction has slight performance gains over them as our scheme requires neither a

[2] Here we focus on *deterministic* constructions without use of nonce or randomization. We also focus on *property-preserving* domain extensions. Hence other constructions—for example RMAC [13]—are excluded from our comparison.

prefix-free encoding nor a $2n$-bit compression function. Maurer and Tessaro [21] treated the problem in a different setting, where the *input* size (rather than output) of the primitive was restricted to n bits.

3 Preliminaries

General Notation. We write $x \leftarrow y$ for the assignment of the value y to a variable x. Given a set X, we write $x \xleftarrow{\$} X$ for the operation of selecting an element uniformly at random from the set X and assigning its value to a variable x. The symbol $x \,\|\, y$ represents the concatenation of two strings x and y, $x \oplus y$ the exclusive OR of x and y, and $|x|$ the length of x in bits. For a finite string $M \in \{0,1\}^*$ and a block size b, the length of M in blocks is the quantity $\ell = \lceil \frac{|M|+1}{b} \rceil$. We adopt the notation $M \xleftarrow{b} M\|10^*$ to signify the "canonical" padding procedure, *i.e.*, $M \leftarrow M\|10^{b\ell-|M|-1}$, where ℓ is the length in blocks of the original M (before being padded). Given a padded message M, we use the notation $m[1] \,\|\, \cdots \,\|\, m[\ell] \xleftarrow{b} M$ as a shorthand for partitioning the string M into b-bit blocks and assigning each block value to $m[1], \ldots, m[\ell]$, so that each $m[i]$ is of b-bit length. Also, we write $x \,\|\, y \xleftarrow{a,b} z$ for dividing the string $z \in \{0,1\}^{a+b}$ into $x \in \{0,1\}^a$ and $y \in \{0,1\}^b$. Throughout the paper we fix the output size n and define $\bar{x} \stackrel{\text{def}}{=} x \oplus 1^n 0^{|x|-n}$ for a string x with $|x| \geq n$.

Adversaries and Games. An adversary is a probabilistic algorithm. An adversary A may take inputs or have access to one or more oracles. We write $y \leftarrow A^{\mathcal{O}}(x)$ to mean that the adversary A, given input x and access to oracle $\mathcal{O}$, outputs a value which is assigned to the variable y. The notation $A^{\mathcal{O}}(x) = y$ indicates the event that the output value of the adversary A, taking an input value x and interacting with oracle $\mathcal{O}$, happens to be equal to the value y.

Often it is convenient to describe oracle behavior in a game style. The notation $G(A)$ indicates running A according to the description of G. By $y \leftarrow G(A)$ we mean the operation of running A in game G, and the value returned by game G is assigned to the variable y. Likewise we define $G(A) = y$. Games frequently involve sets and flags. A set Set is used to record certain specific types of values that appear in the game. We write $\mathsf{Set} \xleftarrow{\cup} x$ for the operation $\mathsf{Set} \leftarrow \mathsf{Set} \cup \{x\}$. A flag flag is used to detect some event that happens in the game. By abuse of notation we let flag also denote the event that the flag flag gets set. We write $\overline{\mathsf{flag}}$ for the complement of the event flag. Unless otherwise stated, sets are set to $\emptyset$ and flags are set to 0 at the beginning of each game execution.

MACs. We adopt the standard single-verification model for the notion of MAC security.[3] Succinctly, for a keyed family of functions $f_k : X \to Y$ with $k \in K$ and for an adversary A define

$$\mathrm{Adv}_f^{\mathrm{mac}}(A) \stackrel{\text{def}}{=} \Pr\left[x^\star \text{ is new} \wedge f_k(x^\star) = y^\star \;\middle|\; (x^\star, y^\star) \leftarrow A^{f_k(\cdot)}, k \xleftarrow{\$} K\right]$$

[3] It suffices to consider only single-verification adversaries, because our MACs are deterministic [2].

as the advantage function,[4] where we call a message $x^\star$ *new* if it has not been queried to oracle $f_k(\cdot)$.

PRFs. The notion of PRF says that a keyed family of functions $f_k : X \to Y$ with a random key $k \xleftarrow{\$} K$ should be indistinguishable from a truly random function $g : X \to Y$. Succinctly, define

$$\mathrm{Adv}_f^{\mathrm{prf}}(A) \stackrel{\mathrm{def}}{=} \Pr\left[A^{f_k(\cdot)} = 1 \;\middle|\; k \xleftarrow{\$} K\right] - \Pr\left[A^{g(\cdot)} = 1 \;\middle|\; g \xleftarrow{\$} \{g : X \to Y\}\right].$$

The notion of PRF implies a secure MAC [2].

PROs. Roughly speaking, the notion of indifferentiability [19,5] corresponds to a type of indistinguishability where adversaries are given oracle access not only to the overlying scheme but also to the underlying primitive. Let $F : \{0,1\}^* \to \{0,1\}^n$ be a mode of operation which iterates a random function $f : \{0,1\}^d \to \{0,1\}^n$. Let $\mathcal{S} : \{0,1\}^d \to \{0,1\}^n$ be a (stateful) simulator having access to a random function $\mathcal{F} : \{0,1\}^* \to \{0,1\}^n$. We define

$$\mathrm{Adv}_{F,\mathcal{S}}^{\mathrm{pro}}(A) \stackrel{\mathrm{def}}{=} \Pr\left[A^{F,f} = 1\right] - \Pr\left[A^{\mathcal{F},\mathcal{S}} = 1\right].$$

Here it is important to note that the simulator cannot "observe" the queries that A makes to $\mathcal{F}$.

Resources. We bound resources of adversaries in terms of its time complexity t, the number of queries q and the total length of queries σ in blocks.[5] We write, for example, $\mathrm{Adv}_f^{\mathrm{mac}}(t,q,\sigma) \stackrel{\mathrm{def}}{=} \max_A \mathrm{Adv}_f^{\mathrm{mac}}(A)$, where the max runs over all adversaries, each having a time complexity at most t, making at most q queries, the total length of queries being at most σ blocks. To measure the time complexity of adversaries, we fix a model of computation. The time complexity of an adversary includes its code size and the total time needed to perform its overlying experiment, in particular the time to access its oracles. We write Time_f for the time to perform one computation of f (for fixed-length inputs) and $\mathrm{Mem}_f(\sigma)$ the memory to store σ-many input/output pairs of f. Lastly we note that for the notion of PROs, the resources of the simulator must be also taken into consideration.

4 Definition of the Double-Piped Mode

Now we give a formal definition of our double-piped mode of operation $F_k : \{0,1\}^* \to \{0,1\}^n$ in Fig. 2. Our mode F takes a secret key $k \in \{0,1\}^\kappa$ and a message $M \in \{0,1\}^*$, outputting a tag $\tau \in \{0,1\}^n$. It iterates a single compression function $f_k : \{0,1\}^d \to \{0,1\}^n$ operating with a single key $k \in \{0,1\}^\kappa$. We require that $d \geq 3n$. We refer back to Fig. 1 for pictorial representation.

[4] Throughout the paper the probabilities are defined over all coins of adversaries, oracles and games.

[5] The block length depends on each scheme, and in our construction it is equal to $d - 2n$ bits.

Algorithm $F_k(M)$
101 $M \xleftarrow{d-2n} M\|10^*$; $m[1] \parallel m[2] \parallel \cdots \parallel m[\ell] \xleftarrow{d-2n} M$
102 $v_1[1] \leftarrow 0^n$; $v_2[1] \leftarrow 0^n$
103 **for** $i = 1$ **to** ℓ
104 $\quad x_1[i] \leftarrow v_1[i] \parallel v_2[i] \parallel m[i]$; $x_2[i] \leftarrow \overline{v_1[i]} \parallel v_2[i] \parallel m[i]$
105 $\quad v_1[i+1] \leftarrow f_k\big(x_1[i]\big)$; $v_2[i+1] \leftarrow f_k\big(x_2[i]\big)$
106 **end for**
107 $\tau \leftarrow f_k\big(v_1[\ell+1] \parallel 1^n \parallel v_2[\ell+1] \parallel 0^{d-3n}\big)$
108 **ret** τ

Fig. 2. Definition of our double-piped mode of operation $F_k : \{0,1\}^* \to \{0,1\}^n$

5 MAC-Pr beyond the Birthday Barrier

In this section we prove that the double-piped mode F is an AIL-MAC with security well above the birthday bound, based on the sole assumption that the compression function f is a secure FIL-MAC. First we roughly outline our proof and then proceed to its full description.

5.1 Outline of the MAC-Pr Proof

We convert a forger attacking F into ones attacking f. We start by observing that the success of forging a tag value for F implies at least one of the following three events:

- **Event** forge**:** A forgery of the tag value for f occurs at the finalization step,
- **Event** ones**:** An output value of f happens to be equal to 1^n at some internal invocation to f, or
- **Event** match**:** The $2n$-bit chaining values at the input to the finalization step happen to be the same for two different queries.

The first two events immediately give us forgers attacking f without birthday degradation, but the third one does not. So we break the third event down further, showing that the match event points to at least one of the following three events:

- **Event** zeros**:** At some iteration point, the $2n$-bit chaining value happens to be equal to 0^{2n},
- **Event** twofold**:** At some iteration point, the output of the "upper" f happens to be equal to that of the "lower" f, or
- **Event** dblcoll**:** A collision of $2n$-bit chaining values occurs somewhere, yielding two collisions of such a type for f.

The zeros and twofold events instantly provide forgers attacking f without birthday degradation. To treat the dblcoll event, however, we need to partition

the case depending on "how many" (single, n-bit) collisions for f are formed in the game. We introduce a threshold value θ and denote by theta the event that "θ-many" (single, n-bit) collisions for f are found. We divide the case as follows:

- **Case** theta**:** In this case we construct a forger attacking f by utilizing the accumulation of collisions. Usually the transformation of a collision into a forgery would lead to birthday degradation [1], but we succeed in maintaining a forgery probability above the birthday bound owing to the large (θ-many) number of collisions.
- **Case** $\mathsf{dblcoll} \wedge \overline{\mathsf{theta}}$**:** On the other hand, if there are not "so many" collisions, then we can create a forger attacking f with a good success probability by utilizing the double, $2n$-bit collision(s). Namely, we first detect that a collision occurs at the half n-bit value, say $v \in \{0,1\}^n$, of the chaining variable (and there are not so many such collisions). We then try to forge a tag, hoping that the remaining half n-bit value will also collide. The (predicted) tag value is chosen from previous chaining variables with its half value colliding to v (and there are not so many candidates for the tag value).

Finally, we choose the threshold value θ appropriately so that the security bound becomes optimal in our setting. This gives us the desired security of $O\big(2^{5n/6}\big)$ query complexity.

5.2 Detailed Proof of MAC-Pr

We now state our main theorem:

Theorem 1. *Let F be the double-piped mode. If the underlying compression function f is a secure FIL-MAC, then the mode F yields a secure AIL-MAC with security above the birthday bound. Specifically, we have*

$$\mathrm{Adv}^{\mathrm{mac}}_{F}(t,q,\sigma) \le 9 \cdot \sigma^{6/5} \cdot \mathrm{Adv}^{\mathrm{mac}}_{f}(t',q+2\sigma),$$

where $t' = t + (q+2\sigma)\cdot \mathrm{Time}_f + \mathrm{Mem}_f(2\sigma)$.

Proof. Let A be a forger attacking F, having a time complexity at most t, making at most q queries to $F_k(\cdot)$ oracle, the query complexity being at most σ blocks. We consider game G_1 as defined in Fig. 3. Game G_1 precisely corresponds to the game defining $\mathrm{Adv}^{\mathrm{mac}}_{F}(A)$, except that it introduces three flags forge, ones and match.

We claim that a successful forgery by A implies at least one of the events forge, ones or match. A forgery by A immediately implies the event forge as long as the value u at line 443 is new. If the value u is not new, then it must have been inserted into the set $\mathsf{Dom}(f)$ at lines 331, 345 or 431. An insertion at line 331 or 431 implies the event ones, while that at line 345 implies match. Thus we have

$$\begin{aligned}\mathrm{Adv}^{\mathrm{mac}}_{F}(A) \stackrel{\text{def}}{=} \Pr\big[G_1(A)=1\big] &\le \Pr\big[\mathsf{forge} \vee \mathsf{ones} \vee \mathsf{match}\big] \\ &\le \Pr\big[\mathsf{forge}\big] + \Pr\big[\mathsf{ones}\big] + \Pr\big[\mathsf{match} \wedge \overline{\mathsf{ones}}\big].\end{aligned}$$

Game $G_1(A)$:
201 $k \xleftarrow{\$} \{0,1\}^\kappa$
202 $(M^\star, \tau^\star) \leftarrow A^{F_k(\cdot)}$
203 **if** $M^\star \in \mathsf{Dom}(F)$ **then ret** 0 **end if**
204 **ret** $V_k(M^\star, \tau^\star)$

On query M **to** $F_k(\cdot)$:
300 $\mathsf{Dom}(F) \xleftarrow{\cup} M$; $M \xleftarrow{d-2n} M\|10^*$
301 $m[1] \parallel m[2] \parallel \cdots \parallel m[\ell] \xleftarrow{d-2n} M$
302 $v_1[1] \leftarrow 0^n$; $v_2[1] \leftarrow 0^n$
303 **for** $i = 1$ **to** ℓ
304 $\quad x_1[i] \leftarrow v_1[i] \parallel v_2[i] \parallel m[i]$
305 $\quad x_2[i] \leftarrow \overline{v_1[i]} \parallel v_2[i] \parallel m[i]$
306 $\quad v_1[i+1] \leftarrow f_k\big(x_1[i]\big)$
▶
311 $\quad v_2[i+1] \leftarrow f_k\big(x_2[i]\big)$
▶
▷
331 $\quad \mathsf{Dom}(f) \xleftarrow{\cup} x_1[i], x_2[i]$
332 $\quad$ **if** $v_1[i+1] = 1^n$ **or** $v_2[i+1] = 1^n$
333 $\quad\quad$ **then** ones $\leftarrow 1$ **end if**
334 **end for**
341 $\mathsf{Dbl} \xleftarrow{\cup} v_1[\ell+1] \parallel v_2[\ell+1]$
343 $u \leftarrow v_1[\ell+1] \parallel 1^n \parallel v_2[\ell+1] \parallel 0^{d-3n}$
345 $\tau \leftarrow f_k(u)$; $\mathsf{Dom}(f) \xleftarrow{\cup} u$
347 **ret** τ

Subroutine $V_k(M, \tau)$:
400 $M \xleftarrow{d-2n} M\|10^*$
401 $m[1] \parallel m[2] \parallel \cdots \parallel m[\ell] \xleftarrow{d-2n} M$
402 $v_1[1] \leftarrow 0^n$; $v_2[1] \leftarrow 0^n$
403 **for** $i = 1$ **to** ℓ
404 $\quad x_1[i] \leftarrow v_1[i] \parallel v_2[i] \parallel m[i]$
405 $\quad x_2[i] \leftarrow \overline{v_1[i]} \parallel v_2[i] \parallel m[i]$
406 $\quad v_1[i+1] \leftarrow f_k\big(x_1[i]\big)$
▶
411 $\quad v_2[i+1] \leftarrow f_k\big(x_2[i]\big)$
▶
▷
431 $\quad \mathsf{Dom}(f) \xleftarrow{\cup} x_1[i], x_2[i]$
432 $\quad$ **if** $v_1[i+1] = 1^n$ **or** $v_2[i+1] = 1^n$
433 $\quad\quad$ **then** ones $\leftarrow 1$ **end if**
434 **end for**
441 **if** $v_1[\ell+1] \parallel v_2[\ell+1] \in \mathsf{Dbl}$
442 $\quad$ **then** match $\leftarrow 1$ **end if**
443 $u \leftarrow v_1[\ell+1] \parallel 1^n \parallel v_2[\ell+1] \parallel 0^{d-3n}$
444 $\tau' \leftarrow f_k(u)$
445 **if** $\tau = \tau'$ **and** $u \notin \mathsf{Dom}(f)$
446 $\quad$ **then** forge $\leftarrow 1$ **end if**
447 **if** $\tau = \tau'$ **then ret** 1 **end if**
448 **ret** 0

Fig. 3. Definition of game $G_1(A)$ for MAC-Pr. The marks ▶ and ▷ indicate that more lines will be inserted in later games.

We start with bounding $\Pr\big[\mathsf{forge}\big]$. We construct a forger B_1 attacking f as follows: B_1 runs the adversary A, simulating $F_k(\cdot)$ oracle and computing the subroutine $V_k(\cdot,\cdot)$ by making queries to its $f_k(\cdot)$ oracle. The adversary B_1 stops at line 444 before making the query u to its $f_k(\cdot)$ oracle and submits a forgery $(u, \tau^\star)$. Note that B_1 always succeeds under the event forge, making at most $q + 2\sigma$ queries to its oracle. So we get

$$\Pr\big[\mathsf{forge}\big] \le \mathrm{Adv}^{\mathrm{mac}}_f(B_1) \le \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma),$$

where $t_1 = t + (q + 2\sigma) \cdot \mathrm{Time}_f$.

We next treat the term $\Pr\big[\mathsf{ones}\big]$. We construct a forger B_2 attacking f as follows: B_2 first picks an index $\alpha \xleftarrow{\$} \{1, 2, \ldots, 2\sigma\}$ and then starts running the adversary A, simulating $F_k(\cdot)$ oracle and (if necessary) computing $V_k(\cdot,\cdot)$ by making queries to its $f_k(\cdot)$ oracle. The adversary B_2 keeps a counter, which is initialized to 0 and gets incremented as B_2 makes a call to $f_k(\cdot)$ oracle except for the finalization step at line 345 where the counter remains unchanged. Just before making the α-th query x_α to $f_k(\cdot)$ oracle, B_2 quits running A and outputs a forgery $(x_\alpha, 1^n)$.

Insert following lines into game G_1 (at mark $\triangleright$):

On query M **to** $F_k(\cdot)$:

320 **if** $v_1[i+1] = v_2[i+1]$
321 **then** twofold $\leftarrow 1$ **end if**
322 $w \leftarrow v_1[i+1] \,\|\, v_2[i+1]$
323 **if** $x_1[i] \notin \mathsf{Dom}_1(f)$ **and** $w \in \mathsf{Dbl}$
324 **then** dblcoll $\leftarrow 1$ **end if**
325 $\mathsf{Dom}_1(f) \xleftarrow{\cup} x_1[i]$; $\mathsf{Dbl} \xleftarrow{\cup} w$
326 **if** $w = 0^{2n}$ **then** zeros $\leftarrow 1$ **end if**

Subroutine $V_k(M, \tau)$:

422 $w \leftarrow v_1[i+1] \,\|\, v_2[i+1]$
423 **if** $x_1[i] \notin \mathsf{Dom}_1(f)$ **and** $w \in \mathsf{Dbl}$
424 **then** dblcoll $\leftarrow 1$ **end if**
425 $\mathsf{Dom}_1(f) \xleftarrow{\cup} x_1[i]$; $\mathsf{Dbl} \xleftarrow{\cup} w$
426 **if** $w = 0^{2n}$ **then** zeros $\leftarrow 1$ **end if**

Fig. 4. Definition of game $G_2(A)$ for MAC-Pr

We now argue that the forger B_2 always succeeds as long as the index α is correctly guessed; the α-th query is expected to be the *first* call to $f_k(\cdot)$ oracle such that the value 1^n gets returned. We verify that the value x_α is new if the index α is such a value. The only thing to check is whether x_α has been already queried at the finalization step (at line 345). If it had been queried, then it would mean that x_α is of the form $*1^n*$ in which the leftmost $*$ is some n-bit string, contradicting with the minimality of α. Now the choice of α is independent of the event ones, so we get $\mathrm{Adv}_f^{\mathrm{mac}}(B_2) \geq \frac{1}{2\sigma} \cdot \Pr[\mathsf{ones}]$. Observe that the adversary B_2 makes at most $q + 2\sigma$ queries to its oracle, and hence we obtain

$$\Pr\big[\mathsf{ones}\big] \leq 2\sigma \cdot \mathrm{Adv}_f^{\mathrm{mac}}(B_2) \leq 2\sigma \cdot \mathrm{Adv}_f^{\mathrm{mac}}(t_1, q + 2\sigma).$$

We proceed to the evaluation of $\Pr\big[\mathsf{match} \wedge \overline{\mathsf{ones}}\big]$. To do this, we consider game G_2 defined in Fig. 4. Game G_2 adds three new flags zeros, twofold and dblcoll to game G_1.

We show that match implies at least one of the events zeros, twofold or dblcoll. Let $(M^\star, \tau^\star)$ be the forgery output by the adversary A. The event match implies that there exists some previous query M' to $F_k(\cdot)$ oracle such that $F_k(M') = F_k(M^\star)$, making an internal collision of the value u at lines 343 and 443. If either (i) M' is a suffix of $M^\star$ with $M^\star$ producing a chaining variable of 0^{2n} or (ii) $M^\star$ is such a suffix of M', then the case immediately implies the zeros event. If neither is such a suffix of the other, then the condition $F_k(M') = F_k(M^\star)$ guarantees that there must be an internal collision of $2n$-bit chaining variables. The collision value may be of the form $v\|v \in \{0,1\}^{2n}$ leading to the event twofold, or otherwise we must have the event dblcoll. Therefore, we have

$$\begin{aligned}\Pr\big[\mathsf{match} \wedge \overline{\mathsf{ones}}\big] &\leq \Pr\big[(\mathsf{zeros} \vee \mathsf{twofold} \vee \mathsf{dblcoll}) \wedge \overline{\mathsf{ones}}\big] \\ &\leq \Pr\big[\mathsf{zeros} \wedge \overline{\mathsf{ones}}\big] + \Pr\big[\mathsf{twofold} \wedge \overline{\mathsf{ones}}\big] \\ &\qquad + \Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}}\big].\end{aligned}$$

We bound the probability $\Pr\big[\mathsf{zeros} \wedge \overline{\mathsf{ones}}\big]$. We construct a forger B_3 attacking f as follows: B_3 first picks an index $\alpha \xleftarrow{\$} \{1, 2, \ldots, \sigma\}$ and then starts running the adversary A, simulating game G_2 by making queries to its $f_k(\cdot)$ oracle. The adversary B_3 keeps a counter, which is initialized to 0 and gets incremented by 1

(and not by 2) as B_3 makes two calls to $f_k(\cdot)$ oracle either at lines 306-311 or at lines 406-411. Just before the α-th execution of lines 306-311 or of lines 406-411, in which two queries x^1_α, x^2_α are about to be made, B_3 quits running A and outputs a forgery $(x^1_\alpha, 0^n)$.

The adversary B_3 always succeeds under the event $\mathsf{zeros} \wedge \overline{\mathsf{ones}}$ along with the condition that the index α is correctly guessed (*i.e.*, the α-th execution of lines 306-311 or of lines 406-411 sets the flag zeros for the first time), because the query x^1_α is guaranteed to be new if the value α is minimal (Notice that either $x^1_\alpha = x^1_{\alpha'}$ or $x^1_\alpha = x^2_{\alpha'}$ for some $\alpha' < \alpha$ implies the event zeros at index α'). In particular, the value x^1_α cannot have been queried at the finalization step (at line 345) due to the event $\overline{\mathsf{ones}}$. Now the choice of α is independent of the event $\mathsf{zeros} \wedge \overline{\mathsf{ones}}$, so we get $\mathrm{Adv}^{\mathrm{mac}}_f(B_3) \geq \frac{1}{\sigma} \cdot \Pr[\mathsf{zeros} \wedge \overline{\mathsf{ones}}]$. The adversary B_3 makes at most $q + 2\sigma$ queries to its oracle, which gives us

$$\Pr\left[\mathsf{zeros} \wedge \overline{\mathsf{ones}}\right] \leq \sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_3) \leq \sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma).$$

We then treat the term $\Pr\left[\mathsf{twofold} \wedge \overline{\mathsf{ones}}\right]$. We construct a forger B_4 attacking f as follows: B_4 first picks an index $\alpha \xleftarrow{\$} \{1, 2, \ldots, \sigma - 1\}$ and then starts running the adversary A, simulating game G_2 by making queries to its $f_k(\cdot)$ oracle. The adversary B_4 keeps the same type of counter as the one used by B_3. Just before the α-th execution of lines 306-311, in which two queries x^1_α, x^2_α are about to be made, B_4 quits running A, makes a query $v^1_\alpha \leftarrow f_k(x^1_\alpha)$ and outputs a forgery (x^2_α, v^1_α).

The adversary B_4 always succeeds under the event $\mathsf{twofold} \wedge \overline{\mathsf{ones}}$ provided that the index α is correctly chosen (*i.e.*, the α-th execution of lines 306-311 sets the flag $\mathsf{twofold}$ for the first time), because the query x^2_α must be new if the value α is minimal (Notice that either $x^2_\alpha = x^1_{\alpha'}$ or $x^2_\alpha = x^2_{\alpha'}$ for some $\alpha' < \alpha$ implies the event $\mathsf{twofold}$ at index α'). Also, the value x^2_α cannot have been queried at the finalization step (at line 345) under the event $\overline{\mathsf{ones}}$. Now the choice of α is independent of the event $\mathsf{twofold} \wedge \overline{\mathsf{ones}}$, so we get $\mathrm{Adv}^{\mathrm{mac}}_f(B_4) \geq \frac{1}{\sigma - 1} \cdot \Pr[\mathsf{twofold} \wedge \overline{\mathsf{ones}}]$. The adversary B_4 makes at most $q + 2\sigma$ queries to its oracle, and hence

$$\Pr\left[\mathsf{twofold} \wedge \overline{\mathsf{ones}}\right] \leq (\sigma - 1) \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_4) \leq (\sigma - 1) \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma).$$

We go on to handle the term $\Pr\left[\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}}\right]$. We introduce a threshold value $\theta = \theta(\sigma)$ in game G_3, as defined in Fig. 5. For the moment we just let θ be a certain function of σ. The description of θ is to be determined at the end of the proof. Game G_3 involves three sets All, Coll and Pair. The set All simply stores all input/output pairs (x, v) already computed as $f_k(x) = v$. The set Coll is a subset of All and stores only those pairs (x, v) that are colliding. The set Pair stores colliding pairs (x', x). We define a function $N(\mathsf{All}, v^\star) \stackrel{\mathrm{def}}{=} \left\{(x, v) \;\middle|\; (x, v) \in \mathsf{All}, v = v^\star\right\}$. We see that

Insert following lines into game G_2 (at mark ▶):

On query M **to** $F_k(\cdot)$:
307 $C\big(x_1[i], v_1[i+1]\big)$
312 $C\big(x_2[i], v_2[i+1]\big)$

Subroutine $V_k(M, \tau)$:
407 $C\big(x_1[i], v_1[i+1]\big)$
412 $C\big(x_2[i], v_2[i+1]\big)$

Subroutine $C(x, v)$:
501 $U \leftarrow N(\mathsf{All}, v)$
502 **if** $(x, v) \notin \mathsf{All}$ **and** $U \neq \emptyset$ **then**
503 $\quad \mathsf{Coll} \xleftarrow{\cup} (x, v)$; $\mathsf{Coll} \leftarrow \mathsf{Coll} \cup U$
504 $\quad$ **for each** $(x', v') \in U$
505 $\quad\quad \mathsf{Pair} \xleftarrow{\cup} (x', x)$ **end for**
506 **end if**
507 $\mathsf{All} \xleftarrow{\cup} (x, v)$
508 **if** $|\mathsf{Pair}| \geq \theta(\sigma)$ **then** $\mathsf{theta} \leftarrow 1$ **end if**

Fig. 5. Definition of game $G_3(A)$ for MAC-Pr

$$\begin{aligned}\Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}}\,\big] &= \Pr\big[(\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \mathsf{theta}) \\ &\qquad \vee (\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}})\big] \\ &\leq \Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big] \\ &\qquad + \Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\big].\end{aligned}$$

We evaluate the probability $\Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big]$. We construct a forger B_5 attacking f as follows: B_5 picks two indices $\alpha, \beta \xleftarrow{\$} \{1, 2, \ldots, 2\sigma\}$ such that $\alpha < \beta$. Then B_5 starts running the adversary A, simulating $F_k(\cdot)$ oracle and (if necessary) computing $V_k(\cdot, \cdot)$ by making queries to its $f_k(\cdot)$ oracle. The adversary B_5 keeps the same type of counter as the one used by forger B_2 (counting the number of calls to $f_k(\cdot)$ oracle except at the finalization step). On making the α-th query x_α to $f_k(\cdot)$ oracle, B_5 records the returned value $v_\alpha \leftarrow f_k(x_\alpha)$. The adversary B_5 resumes running A. Then just before making the β-th query x_β to $f_k(\cdot)$, B_5 stops running A and outputs (x_β, v_α) as a forgery.

The adversary B_5 succeeds in forgery if $f_k(x_\alpha) = f_k(x_\beta)$ and the value x_β has not been queried at any previous point $\gamma < \beta$. Note that the query x_β cannot have been made at the finalization step, since we are under the event $\overline{\mathsf{ones}}$. Now we see that the number of such working pairs (α, β) is exactly $|\mathsf{Pair}|$, and this quantity $|\mathsf{Pair}|$ becomes larger than $\theta(\sigma)$ if the event theta occurs. Since there are $\binom{2\sigma}{2}$ choices of (α, β) total and this choice is independent from the event $\overline{\mathsf{ones}} \wedge \mathsf{theta}$, we obtain $\mathrm{Adv}^{\mathrm{mac}}_f(B_5) \geq \frac{\theta(\sigma)}{\binom{2\sigma}{2}} \cdot \Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big]$. This yields

$$\Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big] \leq \frac{\binom{2\sigma}{2}}{\theta(\sigma)} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_5) \leq \frac{2\sigma^2}{\theta(\sigma)} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma),$$

where we observe that B_5 makes at most $q + 2\sigma$ queries to its $f_k(\cdot)$ oracle.

Lastly, we assess the probability $\Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\big]$. For this, it is helpful to give a graphical interpretation of the sets Coll and Pair. See Fig. 6. The vertices of the *collision graph* are simply the points in Coll. Two distinct points (x, v), $(x', v') \in \mathsf{Coll}$ are connected by an edge if they are colliding, *i.e.*, if $v = v'$. Hence, the edges correspond to the points in Pair. The

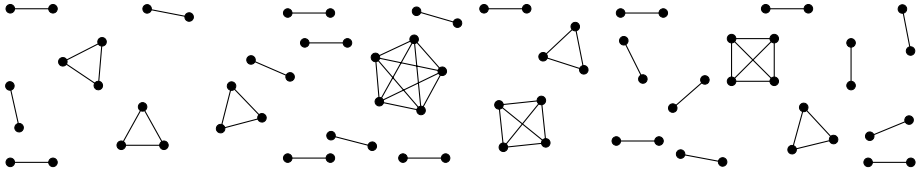


Fig. 6. A simple illustration of the collision graph. The number of vertices is equal to $|\mathsf{Coll}|$, and the number of edges is equal to $|\mathsf{Pair}|$.

collision graph is always a disjoint union of complete graphs, containing no isolated vertices.

Now we construct a forger B_6 attacking f under the event $\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}$. First observe that the event $\overline{\mathsf{theta}}$ sets a limit on the number of vertices in the collision graph. The number is at most $2\theta(\sigma)$, for otherwise the number of edges would exceed $\theta(\sigma)$. Hence we have $|\mathsf{Coll}| \leq 2\theta(\sigma)$ throughout the game. The adversary B_6 first picks an index $\beta \xleftarrow{\$} \{2, 3, \ldots, \lceil 2\theta(\sigma) \rceil\}$ and then starts running A, simulating game G_3 by making queries to its $f_k(\cdot)$ oracle. B_6 keeps a counter, and on the β-th insertion of a colliding value (x_β, v_β) into the set Coll (so that at this point $|\mathsf{Coll}| = \beta$), B_6 stops running A and carries out the following operation:

1. If the β-th insertion happens to be at the "lower" pipe (*i.e.*, at lines 312 or 412), then B_6 aborts.
2. Otherwise, the β-th insertion of (x_β, v_β) occurs at the "upper" pipe (*i.e.*, at lines 307 or 407), in which case B_6 computes the following:
   (a) Choose $(x_\alpha, v_\alpha) \xleftarrow{\$} N(\mathsf{All}, v_\beta) \setminus \{(x_\beta, v_\beta)\}$ so that $x_\alpha \neq x_\beta$ and $f_k(x_\alpha) = v_\alpha = v_\beta = f_k(x_\beta)$.
   (b) Obtain the value $v'_\alpha \leftarrow f_k(\bar{x}_\alpha)$ by either querying $\bar{x}_\alpha$ to $f_k(\cdot)$ oracle again or searching the set All for the entry $(\bar{x}_\alpha, \cdot\,)$.
   (c) Submit $(\bar{x}_\beta, v'_\alpha)$ as a forgery.

We verify that B_6 succeeds in forgery as long as the values for β and x_α are guessed correctly (and that the case B_6 aborts is not a concern). The basic idea is that B_6 hopes to have $(v_\alpha, v'_\alpha) = (v_\beta, v'_\beta)$ with $v'_\beta \stackrel{\text{def}}{=} f_k(\bar{x}_\beta)$ and $\bar{x}_\beta$ being new. The event $\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll}$ guarantees the existence of such a pair (α, β) with $(v_\alpha, v'_\alpha) = (v_\beta, v'_\beta)$, $x_\alpha \neq x_\beta$ and $x_\alpha \neq \bar{x}_\beta$. So let $\alpha^\star < \beta^\star$ be the minimal indices satisfying these conditions (Note that at $(\alpha^\star, \beta^\star)$ the event $\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll}$ does not necessarily occur, since the query $x_{\alpha^\star}$ may well be in the "lower" pipe). The minimality ensures that $\bar{x}_{\beta^\star}$ is new under the event $\overline{\mathsf{ones}}$.

We evaluate the success probability of B_6. In order to do this, we need to count the number of possible values for x_α at step 2(a). We claim that this number is at most $\sqrt{2\theta(\sigma)}$. To see this, observe that the values for x_α come from the vertices of the connected component corresponding to the output collision value v_β. Such a connected component is a complete graph, and the number of vertices must be at

most $\sqrt{2\theta(\sigma)}+1$, for otherwise the number of edges would exceed $\binom{\lceil \sqrt{2\theta(\sigma)}+1 \rceil}{2} \geq \frac{\left(\sqrt{2\theta(\sigma)}+1\right)\sqrt{2\theta(\sigma)}}{2} > \theta(\sigma)$, contradicting with the event $\overline{\mathsf{theta}}$. Hence the number of possible candidates for x_α is at most $\left(\sqrt{2\theta(\sigma)}+1\right)-1 = \sqrt{2\theta(\sigma)}$, excluding the point x_β. Now observe that the choices of β and x_α are completely hidden from the transcript of A, and these values do not affect the probability $\Pr\left[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\right]$, yielding $\mathrm{Adv}_f^{\mathrm{mac}}(B_6) \geq \frac{1}{2\theta(\sigma)} \cdot \frac{1}{\sqrt{2\theta(\sigma)}} \cdot \Pr\left[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\right]$. Hence we get

$$\begin{aligned}\Pr\left[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\right] &\leq 2\theta(\sigma)\sqrt{2\theta(\sigma)} \cdot \mathrm{Adv}_f^{\mathrm{mac}}(B_6) \\ &\leq 3 \cdot \theta(\sigma)^{3/2} \cdot \mathrm{Adv}_f^{\mathrm{mac}}(t_2, q+2\sigma),\end{aligned}$$

where $t_2 = t + (q+2\sigma) \cdot \mathrm{Time}_f + \mathrm{Mem}_f(2\sigma)$. Note that B_6 makes at most $q+2\sigma$ queries to its oracle and maintains the list All, which consumes at most $\mathrm{Mem}_f(2\sigma)$ amount of time complexity.

It remains to determine the threshold function $\theta(\sigma)$. To do this, we sum up the terms obtained:

$$\begin{aligned}\mathrm{Adv}_F^{\mathrm{mac}}(A) \leq\ & \mathrm{Adv}_f^{\mathrm{mac}}(B_1) + 2\sigma \cdot \mathrm{Adv}_f^{\mathrm{mac}}(B_2) \\ &+ \sigma \cdot \mathrm{Adv}_f^{\mathrm{mac}}(B_3) + (\sigma-1) \cdot \mathrm{Adv}_f^{\mathrm{mac}}(B_4) \\ &+ \frac{2\sigma^2}{\theta(\sigma)} \cdot \mathrm{Adv}_f^{\mathrm{mac}}(B_5) + 3 \cdot \theta(\sigma)^{3/2} \cdot \mathrm{Adv}_f^{\mathrm{mac}}(B_6).\end{aligned}$$

Now we simply set $\theta(\sigma) \stackrel{\mathrm{def}}{=} \sigma^{4/5}$. This choice leads to the coefficients of $\frac{2\sigma^2}{\theta(\sigma)} = 2\sigma^{6/5}$ and $3 \cdot \theta(\sigma)^{3/2} = 3\sigma^{6/5}$. Rounding off the terms yields $\mathrm{Adv}_F^{\mathrm{mac}}(A) \leq 9 \cdot \sigma^{6/5} \cdot \mathrm{Adv}_f^{\mathrm{mac}}(t_2, q+2\sigma)$, as desired. □

5.3 On the Tightness of the Bound $O\left(2^{5n/6}\right)$

At the current stage the best attack we know is the birthday attack, which requires $O(2^n)$ query complexity. Hence the bound obtained $O\left(2^{5n/6}\right)$ is not tight. The gap originates in our construction of adversary B_6. Recall that when B_6 makes a choice of β it assumes the worst case scenario of $2\theta(\sigma)$-many vertices, the collision graph being a disjoint union of numerous 2-complete graphs. On the other hand, when B_6 makes a choice of x_α it assumes the other worst case scenario of $\sqrt{2\theta(\sigma)}$-many vertices, the collision graph being a single gigantic complete graph. Hence we are considering two extreme cases which cannot happen concurrently. It remains an open problem to fill in the gap between our proof bound $O\left(2^{5n/6}\right)$ and the best known attack bound $O(2^n)$.

6 PRF-Pr beyond the Birthday Barrier

The PRF-Pr property of our mode immediately follows from the forthcoming PRO-Pr result. This implication is due to the following simple lemma:

Lemma 1 (PRO-Pr ⇒ PRF-Pr in the Dedicated-Key Setting). *Let $F_k : \{0,1\}^* \to \{0,1\}^n$ be a mode of operation in the dedicated-key setting, which iterates a primitive $f_k : \{0,1\}^d \to \{0,1\}^n$ with a secret key k. If the mode F is PRO-Pr in the sense of iterating a random function $f : \{0,1\}^d \to \{0,1\}^n$, then it is PRF-Pr. Specifically, for any simulator $\mathcal{S}$, we have*

$$\mathrm{Adv}^{\mathrm{prf}}_{F}(t,q,\sigma) \le \mathrm{Adv}^{\mathrm{prf}}_{f}(t',q') + \mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(t,q,\sigma),$$

where $t' = t + q' \cdot \mathrm{Time}_f$ and q' is the number of calls to f necessary to process σ-long queries to F.

Proof. Let A be a distinguisher attacking F, having a time complexity at most t, making queries whose total complexity is at most σ blocks. We let $F^\star : \{0,1\}^* \to \{0,1\}^n$ denote the mode of operation identical to F except that the underlying primitive is replaced with a random function $f : \{0,1\}^d \to \{0,1\}^n$ (rather than a pseudo-random function f_k). Let $\mathcal{F} : \{0,1\}^* \to \{0,1\}^n$ be a random function. We can bound the advantage $\mathrm{Adv}^{\mathrm{prf}}_{F}(A)$ as

$$\begin{aligned}
&\Pr\left[A^{F_k(\cdot)} = 1\right] - \Pr\left[A^{\mathcal{F}(\cdot)} = 1\right] \\
&= \Pr\left[A^{F_k(\cdot)} = 1\right] - \Pr\left[A^{F^\star(\cdot)} = 1\right] + \Pr\left[A^{F^\star(\cdot)} = 1\right] - \Pr\left[A^{\mathcal{F}(\cdot)} = 1\right] \\
&\le \mathrm{Adv}^{\mathrm{prf}}_{f}(B) + \mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(A),
\end{aligned}$$

where we construct the distinguisher B, who attacks f, "naturally" from A who is distinguishing between F_k and $F^\star$. Note that $\mathcal{S}$ can be any simulator. We see that B makes at most q' queries to its f oracle and that A makes no queries to the underlying primitive or to the simulator. This gives us the bound. □

7 PRO-Pr beyond the Birthday Barrier

In this section we show that our double-piped mode of operation $F : \{0,1\}^* \to \{0,1\}^n$ is PRO-Pr with security above the birthday bound. It turns out that our mode provides the full PRO security of $O(2^n)$ query complexity. Our proof essentially follows the lines of [3].

In order to state our theorem, we give a description of simulators. See Fig. 7. The simulators maintain a directed, edge-labeled graph structure $\mathsf{Vert}(f)$ and $\mathsf{Edge}(f)$. The symbol $P(\mathsf{Edge}(f), w)$ denotes the set of "paths" $m_1 \| m_2 \| \cdots$ (concatenated labels) starting at the origin 0^{2n} and connecting to the vertex w. The simulators also involve arrays $f[\cdot]$ and $g[\cdot]$, which are everywhere undefined at the beginning of the game. The symbol $\mathsf{Dom}(f)$ denotes the set of already-defined domain points in the array f.

Theorem 2. *The mode of operation F is PRO-Pr beyond the birthday bound. Specifically, we have*

$$\mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(t,q_f,\sigma_F) \le \frac{\sigma_F + q_f}{2^{n-1}} + \frac{5\sigma_F^2 + 18\sigma_F q_f + 17 q_f^2}{2^{2n}},$$

where the simulator $\mathcal{S}$ has a time complexity at most $t + \mathrm{Mem}_f(2\sigma_F)$ and makes at most q_f queries to $\mathcal{F}$ oracle (the idealized F).

Initialization:
600 $\mathsf{Vert}(f) \leftarrow \{0^{2n}\}$

Simulator $\mathcal{S}(x)$:
700 **if** $x \in \mathsf{Dom}(f)$ **then ret** $f[x]$ **end if**
701 $v_1 \| v_2 \| m \xleftarrow{n,n,d-2n} x$
702 **if** $v_2 = 1^n$ **then** $v_2' \| m' \xleftarrow{n,d-3n} m$
703 **if** $v_1 \| v_2' \in \mathsf{Vert}(f)$ **then**
704 $M \xleftarrow{\$} P\big(\mathsf{Edge}(f), v_1 \| v_2'\big)$
705 $f[x] \leftarrow \mathcal{F}(M)$
706 **else** $f[x] \xleftarrow{\$} \{0,1\}^n$ **end if**
707 **else** $f[x] \xleftarrow{\$} \{0,1\}^n$; $f[\bar{x}] \xleftarrow{\$} \{0,1\}^n$
708 **if** $v_1 \| v_2 \in \mathsf{Vert}(f)$ **then**
709 $\mathsf{Vert}(f) \xleftarrow{\cup} f[x] \parallel f[\bar{x}]$
710 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(v_1 \| v_2, m, f[x] \| f[\bar{x}]\big)$
711 **end if**
712 **if** $\bar{v}_1 \| v_2 \in \mathsf{Vert}(f)$ **then**
713 $\mathsf{Vert}(f) \xleftarrow{\cup} f[\bar{x}] \parallel f[x]$
714 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(\bar{v}_1 \| v_2, m, f[\bar{x}] \| f[x]\big)$
715 **end if end if**
716 **ret** $f[x]$

Simulator $\mathcal{S}_f(x)$:
800 **if** $x \in \mathsf{Dom}(f)$ **then ret** $f[x]$ **end if**
801 $w \| m \xleftarrow{2n,d-2n} x$
802 $f[x] \xleftarrow{\$} \{0,1\}^n$; $f[\bar{x}] \xleftarrow{\$} \{0,1\}^n$
803 **if** $w \in \mathsf{Vert}(f)$ **then**
804 $\mathsf{Vert}(f) \xleftarrow{\cup} f[x] \parallel f[\bar{x}]$
805 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(w, m, f[x] \| f[\bar{x}]\big)$ **end if**
806 **if** $\bar{w} \in \mathsf{Vert}(f)$ **then**
807 $\mathsf{Vert}(f) \xleftarrow{\cup} f[\bar{x}] \parallel f[x]$
808 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(\bar{w}, m, f[\bar{x}] \| f[x]\big)$ **end if**
809 **ret** $f[x]$

Simulator $\mathcal{S}_g(u)$:
900 **if** $u \in \mathsf{Dom}(g)$ **then ret** $g[u]$ **end if**
901 **if** $u \in \mathsf{Vert}(f)$ **then**
902 $M \xleftarrow{\$} P\big(\mathsf{Edge}(f), u\big)$
903 $g[u] \leftarrow \mathcal{H}(M)$
904 **else** $g[u] \xleftarrow{\$} \{0,1\}^n$ **end if**
905 **ret** $g[u]$

Fig. 7. Definitions of simulators $\mathcal{S}$, $\mathcal{S}_f$ and $\mathcal{S}_g$ for PRO-Pr

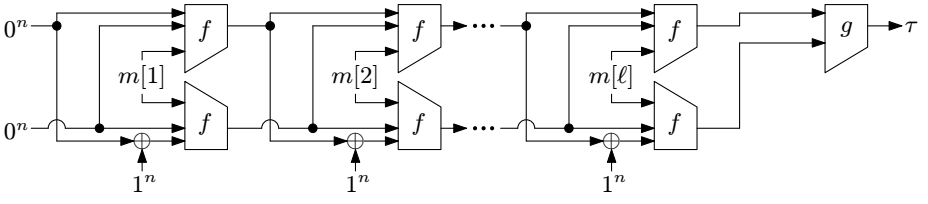


Fig. 8. Description of $H : \{0,1\}^* \to \{0,1\}^n$ iterating two random functions f and g

Proof (Sketch). First we introduce a slightly modified mode of operation $H : \{0,1\}^* \to \{0,1\}^n$. See Fig. 8. The new mode H iterates two independent random functions f and g. The mode H is identical to the original F except that the finalization step is replaced with the new function g.

We start with noting that F is secure if H is secure. The reduction is without birthday degradation. Namely, we show that if H is PRO-Pr beyond the birthday bound, then so is F. More specifically, we obtain

$$\mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(t, q_f, \sigma_F) \le \mathrm{Adv}^{\mathrm{pro}}_{H,\mathcal{S}_f,\mathcal{S}_g}(t, q_f, q_f, \sigma_F) + \frac{\sigma_F + q_f}{2^n},$$

where in the parameters of $\mathrm{Adv}^{\mathrm{pro}}_{H,\mathcal{S}_f,\mathcal{S}_g}$ the second "q_f" is for the number of queries to g oracle and the last σ_F for the total complexity of queries made to H oracle. The proof is the same as the one for Theorem 5.2 in [3].

Now it remains to analyze the security of H. The analysis amounts to bounding the probability that certain "bad" events occur as in [3]. The key to keeping the security bound in $O(2^n)$ is to treat the output values of f always as pairs $\big(f(x), f(\bar{x})\big)$ and to keep track of those "bad" events associated with the $2n$-bit values. By doing so, we are able to obtain

$$\mathrm{Adv}^{\mathrm{pro}}_{H,\mathcal{S}_f,\mathcal{S}_g}(t, q_f, q_g, \sigma_H) \le \frac{\sigma_H + q_f}{2^n} + \frac{5\sigma_H^2 + 14\sigma_H q_f + 12q_f^2 + 4\sigma_H q_g + q_g + 4q_f q_g}{2^{2n}},$$

which yields the claimed bound. □

Acknowledgments

The author is most grateful to the Eurocrypt 2009 anonymous reviewers for their valuable comments. One of the reviewers carried out a thorough review of the MAC-Pr proof and pointed out a couple of typos. Some of the reviewers pointed out an inappropriate treatment of the related work. These comments were especially helpful in revising the paper.

References

1. An, J.H., Bellare, M.: Constructing VIL-mACs from FIL-mACs: Message authentication under weakened assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
2. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive: Report 2004/304 (2004)
3. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
4. Bellare, M., Ristenpart, T.: Hash functions in the dedicated-key setting: Design choices and MPP transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
5. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
6. Chang, D., Lee, S.-J., Nandi, M., Yung, M.: Indifferentiable security analysis of popular hash functions with prefix-free padding. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 283–298. Springer, Heidelberg (2006)
7. Chang, D., Nandi, M.: Improved indifferentiability security analysis of chopMD hash function. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 429–443. Springer, Heidelberg (2008)
8. Coron, J.-S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (2008)

9. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
10. Dodis, Y., Puniya, P.: Feistel networks made public, and applications. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 534–554. Springer, Heidelberg (2007)
11. Dodis, Y., Pietrzak, K., Puniya, P.: A new mode of operation for block ciphers and length-preserving mACs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 198–219. Springer, Heidelberg (2008)
12. Hirose, S., Park, J.H., Yun, A.: A simple variant of the merkle-damgård scheme with a permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
13. Jaulmes, É., Joux, A., Valette, F.: On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 237–251. Springer, Heidelberg (2002)
14. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
15. JTC1: Data cryptographic techniques—Data integrity mechanism using a cryptographic check function employing a block cipher algorithm, ISO/IEC 9797 (1989)
16. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
17. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
18. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
19. Maurer, U.M., Renner, R.S., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
20. Maurer, U.M., Sjödin, J.: Single-key AIL-mACs from any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
21. Maurer, U.M., Tessaro, S.: Domain extension of public random functions: Beyond the birthday barrier. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 187–204. Springer, Heidelberg (2007)
22. NIST: Computer data authentication, FIPS 113 (1985)
23. Patarin, J.: Security of random feistel schemes with 5 or more rounds. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 106–122. Springer, Heidelberg (2004)
24. Patarin, J.: A proof of security in $O(2^n)$ for the Benes scheme. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 209–220. Springer, Heidelberg (2008)
25. Preneel, B., van Oorschot, P.C.: MDx-MAC and building fast mACs from hash functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
26. Preneel, B., van Oorschot, P.C.: On the security of iterated message authentication codes. IEEE Transactions on Information Theory 45(1), 188–199 (1999)
27. Yasuda, K.: Multilane HMAC— security beyond the birthday limit. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 18–32. Springer, Heidelberg (2007)
28. Yasuda, K.: A one-pass mode of operation for deterministic message authentication— security beyond the birthday barrier. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 316–333. Springer, Heidelberg (2008)

On the Security of Cryptosystems with Quadratic Decryption: The Nicest Cryptanalysis

Guilhem Castagnos[1,⋆] and Fabien Laguillaumie[2]

[1] PRISM - Université de Versailles St-Quentin-en-Yvelines
45, avenue des États-Unis, 78035 Versailles Cedex, France
guilhem.castagnos@prism.uvsq.fr
[2] GREYC - Université de Caen-Basse Normandie
Boulevard du Maréchal Juin, BP 5186, 14032 Caen Cedex, France
fabien.laguillaumie@info.unicaen.fr

Abstract. We describe the first *polynomial time chosen-plaintext total break* of the NICE family of cryptosystems based on ideal arithmetic in imaginary quadratic orders, introduced in the late 90's by Hartmann, Paulus and Takagi [HPT99]. The singular interest of these encryption schemes is their natural quadratic decryption time procedure that consists essentially in applying Euclid's algorithm. The only current specific cryptanalysis of these schemes is Jaulmes and Joux's chosen-ciphertext attack to recover the secret key [JJ00]. Originally, Hartmann *et al.* claimed that the security against a total break attack relies *only* on the difficulty of factoring the public discriminant $\Delta_q = -pq^2$, although the public key was also composed of a specific element of the class group of the order of discriminant Δ_q, which is crucial to reach the quadratic decryption complexity. In this article, we propose a drastic cryptanalysis which factors Δ_q (and hence recovers the secret key), only given this element, in cubic time in the security parameter. As a result, performing our cryptanalysis on a cryptographic example takes less than a second on a standard PC.

Keywords: Polynomial time total break, quadratic decryption, NICE cryptosystems, imaginary quadratic field-based cryptography.

1 Introduction

We propose an original and radical cryptanalysis of a large class of schemes designed within imaginary quadratic fields, based on the NICE cryptosystem (cf. [HPT99,PT99,PT00]) which recovers the secret key from the sole public key. These systems have been intensively developed and studied in the late 90's, since they offer a very efficient secret operation (decryption or signature), compared to cryptosystems based on traditional number theory. The one-wayness of these

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

⋆ This work was done while this author was with the GREYC - ENSICAEN.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 260–277, 2009.

schemes rely on the difficulty of the *Smallest Kernel-Equivalent Problem* (SKEP) and their security against a total break was believed to rely on the difficulty of the factorisation of numbers of the form pq^r. The first and only cryptanalysis of the NICE encryption scheme, proposed by Jaulmes and Joux's at Eurocrypt'00 [JJ00], recovers the secret key with an access to a decryption oracle[1]. In the setting of the NICE cryptosystems, the public key contains a discriminant $\Delta_q = -pq^2$ *and* the representation of a reduced ideal $\mathfrak{h}$ whose class belongs to the kernel of the surjection from the class group of the quadratic order of (public) discriminant $\Delta_q = -pq^2$ to the class group of the maximal order of (secret) discriminant $\Delta_K = -p$. We will show that with this knowledge of $\mathfrak{h}$ we can actually factor the public discriminant in cubic time in the security parameter.

1.1 Imaginary Quadratic Field-Based Cryptography

The first use of class groups of imaginary quadratic fields allowed to achieve a Diffie-Hellman key exchange. This paper by Buchmann and Williams [BW88] was the first of several attempts to design imaginary quadratic field-based cryptosystems. Key exchange was also discussed by McCurley in [McC89]. Ten years after, a new encryption scheme appeared in the literature, in the work of Hühnlein, Jacobson, Paulus and Takagi [HJPT98]. The goal of this paper was also to improve the efficiency of the seminal cryptosystems. In fact, the key point of these Elgamal-like encryption schemes is the switching between the class group of the maximal order and the class group of a non-maximal order, which can be done with quadratic complexity (as already mentioned). Unfortunately, Hühnlein *et al.*'s scheme, although using this efficient switching, did not benefit from a quadratic time decryption since the decryption of this scheme really needed a final exponentiation (like in Elgamal).

Soon after, quadratic decryption time was eventually reached with a new encryption scheme, called *NICE*, for *New Ideal Coset Encryption*, described in [HPT99, PT99, PT00]. In [HPT99], it is shown that the decryption time of NICE is comparably as fast as the encryption time of RSA with public exponent $e = 2^{16} + 1$ and an even better implementation is described by Hühnlein in [Huh00]. The key idea of NICE is not to mask the message by a power of the public key (which leads to a cubic decryption like in Elgamal), but by an element which belongs to the kernel of the map which switches between the class group of a non-maximal order to the maximal order. This hiding element is added to the public key and naturally disappears from the ciphertext when applying the map.

As the semantic security of NICE holds only under a chosen-plaintext attack, Buchmann, Sakurai and Takagi patched the scheme by adapting classical techniques to obtain a chosen-ciphertext security in the random oracle model [BST02]. This enhanced scheme, based on REACT [OP01] is called NICE-X, and of course resists Jaulmes and Joux's attack [JJ00]. Hühnlein, Meyer and Takagi also built in [HMT99] Rabin and RSA analogues based on non-maximal imaginary quadratic orders, but the only advantages over the original systems

[1] This attack can actually be deflected by adding a suitable padding.

is their seemingly natural immunity against low exponent attacks and some chosen-ciphertext attacks.

The design of signature schemes has also been addressed in [HM00, Huh01] with an adaptation of Schnorr signatures (cf. [Sch00]). Again an element of the kernel of the switching between two class groups is published: this element is crucial for the efficiency of the signature generation. An undeniable signature scheme has been designed in [BPT04], and again, the public element of the kernel is needed for the design of an efficient scheme.

1.2 Related Work on Security Issues of Quadratic Field-Based Cryptography

All the NICE schemes share the same public information: a discriminant of the form $\Delta_q = -pq^2$ and the representation of a reduced ideal $\mathfrak{h}$ whose class belongs to the kernel of the surjection from the class group of the quadratic order of (public) discriminant $\Delta_q = -pq^2$ to the class group of the maximal order of (secret) discriminant $\Delta_K = -p$. Of course, a factorisation of the discriminant obviously totally breaks the scheme. Therefore, the security parameters are set such that the factorisation of numbers of the form pq^r is difficult. This particular factorisation has been addressed by Boneh, Durfee and Howgrave-Graham in [BDH99], but for small r (such as 2), their method is not better than Lenstra's ECM method [Len87] or the Number Field Sieve [LL93]. In [BST02], the authors also mention the *Quadratic Order Discrete Logarithm Problem* (QODLP). The fastest algorithm to solve the QODLP is the Hafner-McCurley algorithm [HM89], but its running time has a worse subexponential complexity than the fastest factoring algorithm. In [PT00], Paulus and Takagi argue that "the knowledge of $\mathfrak{h}$ does not substantially help to factor Δ_q using currently known fast algorithms". They also mention the possibility to find a power of the class $[\mathfrak{h}]$ of order 2, but computing the order of the class $[\mathfrak{h}]$ in the class group of the order of discriminant Δ_q is essentially equivalent to factor this discriminant. The problem of factoring the discriminant Δ_q given $[\mathfrak{h}]$ is called the Kernel Problem in [BPT04] and again is assumed to be "intractable".

Up to now, the sole specific cryptanalysis of this family of encryption schemes is the *chosen-ciphertext* nice cryptanalysis from [JJ00]. This attack uses the fact that the decryption fails (*i.e.*, does not recover the plain message) if the norm of the ideal representing the message is greater than $\sqrt{|\Delta_K|/3}$, so that the decoded message will expectedly be one step from being reduced. The relation between two pairs original message/decoded message leads to a Diophantine equation of the form $k = XY$ for a known "random" integer k of the size of the secret primes. The authors suggest to factor this integer to find out X and Y and then factor Δ_q. This attack is feasible for the parameters proposed in [HPT99], but can be defeated by enlarging the key size by a factor of 3. No complexity analysis is given for this attack, and the scheme can also be repaired by adding redundancy to the message as suggested in [JJ00] and [BST02]. Note that, contrary to ours, Jaulmes and Joux's attack also applies to [HJPT98].

1.3 Our Contributions

We propose the first definitive cryptanalysis of cryptosystems based on NICE, which have been resisting for almost 10 years. All these schemes contain in the public key the representation of the reduced ideal $\mathfrak{h}$ whose class belongs to the kernel of the surjection from the class group of the quadratic order of discriminant $\Delta_q = -pq^2$ to the class group of the maximal order of discriminant $\Delta_K = -p$. The key point of our attack is the fact that this ideal $\mathfrak{h}$ is indeed always equivalent to a non-reduced ideal of norm q^2, as we will show in Theorem 2. The core of our attack then consists of lifting the class of $\mathfrak{h}$ in the class group of the order of discriminant $\Delta_q r^2$, where r is chosen to make the ideals of norm q^2 reduced. This operation will reveal an ideal of norm q^2 and thus the factorisation of Δ_q, leading to a total break of the scheme.

Note that the public ideal $\mathfrak{h}$ is crucial in the design of NICE: Random powers of this element are used to hide the message. As it is in the kernel of a surjective map, this randomness can be removed from the ciphertext and the message recovered by applying this map which leads to a decryption algorithm with quadratic complexity (the computation is done with Euclid's algorithm).

The attack described in this paper thus uses this extra piece of information given in the public key to factor the public discriminant. Therefore, this setting is insecure in order to build a cryptosystem with quadratic decryption time. Note that such a scheme with quadratic decryption is a very rare object in group theory based cryptography. Although some schemes built from lattices or coding theory problems have this property, to our knowledge, very few schemes built from the integer factorisation or the discrete logarithm problems have it (*e. g.*, variants of Okamoto-Uchiyama and Paillier's cryptosystems, cf. [CNP99,Pai99]).

As a matter of fact, the encryption schemes built on NICE from [HPT99,PT99, PT00,BST02,Huh00], the signature schemes [Huh01,HM00] and the undeniable signature scheme [BPT04] totally succumb to our attack.

The rest of the paper is organised as follows: The next section gives a background on orders of imaginary quadratic fields to understand the NICE cryptosystem, and then Section 3 is the core of the paper. We describe the cryptanalysis by first discussing the (im-)possibility of reversing the reduction process applied on the reduced ideal $\mathfrak{h}$ in Subsection 3.1. Then, in Subsection 3.2, we describe our attack (Algorithm 3) whose correctness is then proved with Theorem 3 and Corollary 1. Finally, we illustrate the attack with an example.

2 Background

The next subsection widely follows the description from [Cox99].

2.1 Computations in Quadratic Orders

A *quadratic field* K is a subfield of the field of complex numbers $\mathbb{C}$ which has degree 2 over $\mathbb{Q}$. Such a field can be uniquely written as $\mathbb{Q}(\sqrt{n})$ where n is a

square-free integer, different from 1 and 0. Its (fundamental) *discriminant* Δ_K is defined as n if $n \equiv 1 \pmod 4$ and $4n$ otherwise. We will then consider K in terms of its discriminant : $K = \mathbb{Q}(\sqrt{\Delta_K})$ with $\Delta_K \equiv 0, 1 \pmod 4$. An *order* $\mathcal{O}$ in K is a subset of K such that $\mathcal{O}$ is a subring of K containing 1 and $\mathcal{O}$ is a free $\mathbb{Z}$-module of rank 2. The ring $\mathcal{O}_{\Delta_K}$ of integers[2] in K is the *maximal* order of K in the sense that it contains all the other orders of K. It can be written as $\mathbb{Z} + \frac{1}{2}(\Delta_K + \sqrt{\Delta_K})\mathbb{Z}$. If we set $f = [\mathcal{O}_{\Delta_K} : \mathcal{O}]$ the *finite* index of any order $\mathcal{O}$ in $\mathcal{O}_{\Delta_K}$, then $\mathcal{O} = \mathbb{Z} + f\frac{1}{2}(\Delta_K + \sqrt{\Delta_K})\mathbb{Z} = \mathbb{Z} + f\mathcal{O}_{\Delta_K}$. The integer f is called the *conductor* of $\mathcal{O}$. The discriminant of $\mathcal{O}$ is then $\Delta_f = f^2\Delta_K$. We will then use the notation $\mathcal{O}_{\Delta_f}$ for such an order.

Now we discuss the ideals of an order $\mathcal{O}_\Delta$ of discriminant Δ. If $\mathfrak{a}$ is a nonzero ideal of $\mathcal{O}_\Delta$, its norm is defined as $\mathrm{N}(\mathfrak{a}) = |\mathcal{O}_\Delta/\mathfrak{a}|$. An ideal $\mathfrak{a}$ is said to be *proper* if $\{\beta \in K : \beta\mathfrak{a} \subset \mathfrak{a}\} = \mathcal{O}_\Delta$. This definition can be extended to *fractional* ideals, which are of the form $\alpha\mathfrak{a}$ where $\alpha \in K^\times$ and $\mathfrak{a}$ is an ideal of $\mathcal{O}_\Delta$. If we denote by $I(\mathcal{O}_\Delta)$ the set of proper fractional ideals of $\mathcal{O}_\Delta$ and its subgroup $P(\mathcal{O}_\Delta)$ consisting of principal ideals, the *ideal class group* of $\mathcal{O}_\Delta$ is defined as $C(\mathcal{O}_\Delta) = I(\mathcal{O}_\Delta)/P(\mathcal{O}_\Delta)$. Its cardinality is the *class number* of $\mathcal{O}_\Delta$ denoted as $h(\mathcal{O}_\Delta)$.

Every ideal $\mathfrak{a}$ of $\mathcal{O}_\Delta$ can be written as

$$\mathfrak{a} = m\left(a\mathbb{Z} + \frac{-b + \sqrt{\Delta}}{2}\mathbb{Z}\right)$$

with $m \in \mathbb{Z}$, $a \in \mathbb{N}$ and $b \in \mathbb{Z}$ such that $b^2 \equiv \Delta \pmod{4a}$. In the sequel, we will only consider *primitive* ideals, which are those with $m = 1$. This expression is unique if $-a < b \leq a$ and we will now denote a primitive ideal by (a, b). The norm of such an ideal is then a.

This notation represents also the positive definite binary quadratic form $ax^2 + bxy + cy^2$ with $b^2 - 4ac = \Delta$. Theorem 7.7 from [Cox99] shows that, up to equivalence relations, it is essentially equivalent to work with ideals and positive definite quadratic forms. An ideal (a, b) of $\mathcal{O}_\Delta$ is said to be *reduced* if the corresponding quadratic form is reduced, which means that $|b| \leq a \leq c$ and $b \geq 0$ if one of the inequalities is not strict. Note that in every class of $\mathcal{O}_\Delta$-ideals there exists exactly one reduced ideal. From the theory of quadratic forms, we can efficiently compute a reduced equivalent ideal. The algorithm, which is due to Gauss, is described in [Coh00, Algorithm 5.4.2 p. 243] and is called Red in the rest of the paper. In general, instead of working with classes, we will work with reduced ideals. The product of ideals is also efficiently computable with the composition of quadratic forms algorithm, see [Coh00, Algorithm 5.4.7 p. 243]. These two algorithms have *quadratic* complexity. A crucial fact for our purpose is described in Lemma 5.3.4 from [Coh00]: If an ideal $\mathfrak{a} = (a, b)$ is reduced, then $a \leq \sqrt{\Delta/3}$ and conversely, if $a < \sqrt{\Delta/4}$ and $-a < b \leq a$, then $\mathfrak{a}$ is reduced.

Let $\left(\frac{a}{b}\right)$ be the Kronecker symbol of a and b. The formula for the class number is given by the following theorem.

[2] *i.e.*, the set of all $\alpha \in K$ which are roots of a monic polynomial in $\mathbb{Z}[X]$

Theorem 1 ([Cox99, Theorem 7.24]). *Let $\mathcal{O}_{\Delta_f}$ be the order of conductor f in an imaginary quadratic field K (i. e., $\Delta_f = f^2\Delta_K$). Then*

$$h(\mathcal{O}_{\Delta_f}) = \frac{h(\mathcal{O}_{\Delta_K})f}{[\mathcal{O}_{\Delta_K}^\times : \mathcal{O}_{\Delta_f}^\times]} \prod_{p|f} \left(1 - \left(\frac{\Delta_K}{p}\right)\frac{1}{p}\right).$$

Given an order $\mathcal{O}_{\Delta_f}$ of conductor f, a nonzero $\mathcal{O}_{\Delta_f}$-ideal $\mathfrak{a}$ is said to be *prime to f* if $\mathfrak{a} + f\mathcal{O}_{\Delta_f} = \mathcal{O}_{\Delta_f}$ (it is equivalent to say that its norm $\mathrm{N}(\mathfrak{a})$ is prime to f – see Lemma 7.18 from [Cox99]). We denote by $I(\mathcal{O}_{\Delta_f}, f)$ the subgroup of $I(\mathcal{O}_{\Delta_f})$ generated by ideals prime to f. $P(\mathcal{O}_{\Delta_f}, f)$ is the subgroup generated by the principal ideals $\alpha\mathcal{O}_{\Delta_f}$ where $\alpha \in \mathcal{O}_{\Delta_f}$ has a norm prime to f. Note that in every ideal class, there exists an ideal prime to f (cf. [Cox99, Corollary 7.17]). To establish Theorem 1, Cox has studied the links between the class group of the maximal order of an imaginary quadratic field and the class groups of any of its orders. The following propositions throw a light on such fundamental links.

Proposition 1 ([Cox99, Proposition 7.19]). *The inclusion $I(\mathcal{O}_{\Delta_f}, f) \subset I(\mathcal{O}_{\Delta_f})$ induces an isomorphism*

$$I(\mathcal{O}_{\Delta_f}, f)/P(\mathcal{O}_{\Delta_f}, f) \simeq I(\mathcal{O}_{\Delta_f})/P(\mathcal{O}_{\Delta_f}) = C(\mathcal{O}_{\Delta_f}).$$

Proposition 2 ([Cox99, Proposition 7.20]). *Let $\mathcal{O}_{\Delta_f}$ be an order of conductor f in an imaginary quadratic field K.*

i. If $\mathfrak{A}$ is an $\mathcal{O}_{\Delta_K}$-ideal prime to f, then $\mathfrak{A} \cap \mathcal{O}_{\Delta_f}$ is an $\mathcal{O}_{\Delta_f}$-ideal prime to f of the same norm.
ii. If $\mathfrak{a}$ is an $\mathcal{O}_{\Delta_f}$-ideal prime to f, then $\mathfrak{a}\mathcal{O}_{\Delta_K}$ is an $\mathcal{O}_{\Delta_K}$-ideal prime to f of the same norm.
iii. The map $\varphi_f : I(\mathcal{O}_{\Delta_f}, f) \longrightarrow I(\mathcal{O}_{\Delta_K}, f)$ such that $\mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_K}$ is an isomorphism.

Consequently, the map φ_f from Proposition 2 induces a surjection

$$\bar{\varphi}_f : C(\mathcal{O}_{\Delta_f}) \longrightarrow\!\!\!\!\!\rightarrow C(\mathcal{O}_{\Delta_K})$$

that can be computed as follows: given a class $[\mathfrak{a}] \in C(\mathcal{O}_{\Delta_f})$, one finds $\mathfrak{b} \in [\mathfrak{a}]$ such that $\mathfrak{b} \in I(\mathcal{O}_{\Delta_f}, f)$ (see standard techniques [HJPT98, Algorithm 1]) and $\bar{\varphi}_f([\mathfrak{a}]) = [\varphi_f(\mathfrak{b})] = [\mathfrak{b}\mathcal{O}_{\Delta_K}]$. The next two algorithms compute φ_f and its inverse (cf. [PT00]).

Input: $\mathfrak{A} = (A, B) \in I(\mathcal{O}_{\Delta_K}, f)$
Output: $\mathfrak{A} \cap \mathcal{O}_{\Delta_f} = (a, b) \in I(\mathcal{O}_{\Delta_f}, f)$

1. $a \leftarrow A$
2. $b \leftarrow Bf \mod_c 2a$ $(|b| < a)$ [centered euclidean division]
3. **Return** (a, b)

Algorithm 1: Algorithm to compute φ_f^{-1}

Input: $\mathfrak{a} = (a, b) \in I(\mathcal{O}_{\Delta_f}, f), \Delta_f$
Output: $\mathfrak{a}\mathcal{O}_{\Delta_K} = (A, B) \in I(\mathcal{O}_{\Delta_K}, f)$
1. $A \leftarrow a$
2. $\delta \leftarrow \Delta_f \mod 2$
3. Compute u and $v \in \mathbb{Z}$ such that $1 = uf + a\delta v$ [extended Euclidean algorithm]
4. $B \leftarrow bu + a\delta v \mod_c 2a$ $(|B| < a)$ [centered euclidean division]
5. **Return** (A, B)

Algorithm 2: Algorithm to compute φ_f

Takagi and Paulus showed, in Section 4.2 from [PT00], that in the NICE setting the computation of this homomorphism φ_f cannot be done without the knowledge of the prime secret conductor.

The following effective lemma was used in [Cox99] to prove the formula of Theorem 1 by computing the order of $\ker \bar{\varphi_f}$ and by Hühnlein, for example in [Huh00, Huh01] to efficiently compute in $\ker \bar{\varphi_f}$. It also proves the correctness of our attack. Indeed with a well-known system of representatives of $(\mathcal{O}_{\Delta_K}/f\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/f\mathbb{Z})^\times$, we will derive a suitable system of representatives for $\ker \bar{\varphi_f}$, which is essential for the proofs of Theorem 2 and Lemma 2.

Lemma 1. *Let Δ_K be a fundamental negative discriminant, different from -3 and -4, and f a conductor. Then there exists an effective isomorphism*

$$\psi_f : (\mathcal{O}_{\Delta_K}/f\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/f\mathbb{Z})^\times \xrightarrow{\sim} \ker \bar{\varphi_f}.$$

We will denote by $\phi_{\Delta_K}(f) := f \prod_{p|f} \left(1 - \left(\frac{\Delta_K}{p}\right) \frac{1}{p}\right)$ the order of $\ker \bar{\varphi_f}$.

Proof. The proof follows the line of the proof of [Cox99, Proposition 7.22 and Theorem 7.24]. □

Remark 1. To effectively map a class from $(\mathcal{O}_{\Delta_K}/f\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/f\mathbb{Z})^\times$ to $\ker \bar{\varphi_f}$, one takes a representative $\alpha \in \mathcal{O}_{\Delta_K}$, $\alpha := x + y \frac{\Delta_K + \sqrt{\Delta_K}}{2}$ where $x, y \in \mathbb{Z}$ and $\gcd(\mathrm{N}(\alpha), f) = 1$ (to ensure that α is invertible modulo $f\mathcal{O}_{\Delta_K}$), and computes

$$\psi_f\big([\alpha]\big) = \big[\varphi_f^{-1}(\alpha\mathcal{O}_{\Delta_K})\big],$$

which is in $\ker \bar{\varphi_f}$. In this computation, the representation of $\alpha\mathcal{O}_{\Delta_K}$ can be obtained with [BTW95, Proposition 2.9] and the evaluation of φ_f^{-1} with Algorithm 1.

Conversely, given a class of $\ker \bar{\varphi_f}$ usually represented by its reduced ideal, one finds a representative ideal $\mathfrak{h} \in I(\mathcal{O}_{\Delta_f}, f)$ (with [HJPT98, Algorithm 1]) and computes $\alpha \in \mathcal{O}_{\Delta_K}$ such that $\alpha\mathcal{O}_{\Delta_K} = \varphi_f(\mathfrak{h})$ (φ_f is evaluated with Algorithm 2 and α can be found with [HJW03, Algorithm 1]). Eventually, $\psi_f^{-1}([\mathfrak{h}]) = [\alpha] \in (\mathcal{O}_{\Delta_K}/f\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/f\mathbb{Z})^\times$.

KeyGen(1^λ)**:**

- Let p be a λ-bit prime such that $p \equiv 3 \pmod 4$ and let q be a prime such that $q > \sqrt{p/3}$.
- Set
$$\begin{cases} \Delta_K = -p \\ \Delta_q = \Delta_K q^2 = -pq^2 \end{cases}$$
- Let k and l be the bit lengths of $\lfloor \sqrt{|\Delta_K|/4} \rfloor$ and $q - \left(\frac{\Delta_K}{q}\right)$ respectively.
- Let $[\mathfrak{h}]$ be an element of $\ker \bar{\varphi}_q$, where $\mathfrak{h}$ is a reduced $\mathcal{O}_{\Delta_q}$- ideal.

The public key pk consists of the quadruple $(\Delta_q, \mathfrak{h}, k, l)$, and the secret key sk consists of the pair (p, q).

Encrypt$(1^\lambda, pk, m)$**:**

- A message m is embedded into a reduced $\mathcal{O}_{\Delta_q}$-ideal $\mathfrak{m}$ with $\log_2(N(\mathfrak{m})) < k$.
- Pick randomly $r \in [\![1, 2^{l-1}]\!]$ and compute $\mathfrak{c} = \mathsf{Red}(\mathfrak{m} \times \mathfrak{h}^r)$.

Decrypt$(1^\lambda, sk, \mathfrak{c})$**:** Compute $\varphi_q^{-1}(\mathsf{Red}(\varphi_q(\mathfrak{c}))) = \mathfrak{m}$.

Fig. 1. Description of NICE

2.2 The NICE family

We will now describe in Fig. 1 the original NICE cryptosystem as it is presented in [PT00]. For our purpose, it is only important to concentrate on the key generation which outputs an element $[\mathfrak{h}]$ of $\ker \bar{\varphi}_q$ as a part of the public key. Other encryption schemes which share this key generation can be found in [HPT99, PT00, BST02, Huh00, PT99], and signature schemes in [Huh01, HM00, BPT04]. As already mentioned, all these cryptosystems succumb to our attack.

Underlying Algorithmic Assumptions. The security against a total break (resp. of the one-wayness) of the NICE cryptosystem is proved to rely on the hardness of the following problems:

Definition 1 (Kernel Problem [BPT04]). *Let λ be an integer, p and q be two λ-bit primes with $p \equiv 3 \pmod 4$. Fix a non-fundamental discriminant $\Delta_q = -pq^2$. Given an element $[\mathfrak{h}]$ of $\ker \bar{\varphi}_q$, factor the discriminant Δ_q.*

Definition 2 (Smallest Kernel-Equivalent Problem [BST02, BPT04] (SKEP)). *Let λ be an integer, p and q be two λ-bit primes with $p \equiv 3 \pmod 4$. Fix a non-fundamental discriminant $\Delta_q = -pq^2$. Given an element $[\mathfrak{h}]$ of $\ker \bar{\varphi}_q$ and an element $[\mathfrak{m}] \in C(\mathcal{O}_{\Delta_q})$, compute the ideal with the smallest norm in the equivalence class, modulo the subgroup generated by $[\mathfrak{h}]$, of $[\mathfrak{m}]$.*

It is clear that an algorithm which solves the Kernel Problem also solves the Smallest Kernel-Equivalent Problem. The insecurity of the Kernel Problem will be discussed in the next section.

3 The Cryptanalysis

3.1 Intuition

In the NICE setting, $\Delta_K = -p$, $\Delta_q = \Delta_K q^2$ where p and q are two large primes, and the schemes are totally broken if one can recover p and q from Δ_q. (Un-)fortunately, another piece of information is given in the public key: an ideal $\mathfrak{h}$ whose class belongs to the kernel of $\bar{\varphi}_q$, the surjection from $C(\mathcal{O}_{\Delta_q})$ to $C(\mathcal{O}_{\Delta_K})$. In [PT00] (for example), the authors suppose that no ideal whose class is in $\ker \bar{\varphi}_q$ leaks a factor of the public discriminant Δ_q, except if this element has order 2, but then a subexponential computation is required to find it.

While investigating this assumption, we experimentally found non-reduced ideals of the form (q^2, kq), with k odd and $|k| < q$ whose classes belong to the kernel of $\bar{\varphi}_q$, and which obviously give the factorisation of Δ_q. By using the effective isomorphism of Lemma 1, we actually prove in the next theorem that one can build a representative set of this kernel with ideals of norm q^2.

Theorem 2. *Let Δ_K be a fundamental negative discriminant, different from -3 and -4 and q an odd prime conductor. There exists an ideal of norm q^2 in each nontrivial class of* $\ker \bar{\varphi}_q$.

Proof. Let us recall the effective isomorphism from Lemma 1:

$$\psi_q \colon \left(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K}\right)^\times / \left(\mathbb{Z}/q\mathbb{Z}\right)^\times \xrightarrow{\sim} \ker \bar{\varphi}_q.$$

We are going to build a set of representatives of $\left(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K}\right)^\times / \left(\mathbb{Z}/q\mathbb{Z}\right)^\times$ and apply ψ_q (which can be computed according to Remark 1) to obtain ideals of norm q^2 which are a set of representatives of $\ker \bar{\varphi}_q$.

Let us set $\alpha_k = k + \frac{\Delta_K + \sqrt{\Delta_K}}{2}$ with $k \in \{0, \ldots, q-1\}$. Clearly $\mathrm{N}(\alpha_k) = \left(k + \frac{\Delta_K}{2}\right)^2 - \frac{\Delta_K}{4} = k^2 + \Delta_K k + \frac{\Delta_K(\Delta_K - 1)}{4}$. Consider the following set of representatives of $\left(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K}\right)^\times / \left(\mathbb{Z}/q\mathbb{Z}\right)^\times$:

$$\left\{1\right\} \cup \left\{\alpha_k \text{ with } k \in \{0, \ldots, q-1\},\, \mathrm{N}(\alpha_k) \not\equiv 0 \pmod q\right\},$$

indeed, it is easy to check that all the α_k belong to different classes and that they are in sufficient number: If $\left(\frac{\Delta_K}{q}\right)$ equals 1 (resp. equals 0, resp. equals -1) then the order of the quotient $\left(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K}\right)^\times / \left(\mathbb{Z}/q\mathbb{Z}\right)^\times$ is $1 + (q-2)$ (resp. $1 + (q-1)$, resp. $1 + (q+1)$). We are now going to compute the image of this set by ψ_q in $\ker \bar{\varphi}_q$.

Following the proof of [BTW95, Proposition 2.9], we detail here the computation of $\mathfrak{A}_k = \alpha_k \mathcal{O}_{\Delta_K}$. The representation of $\mathfrak{A}_k$ is (a_k, b_k), with $a_k = \mathrm{N}(\alpha_k)$. Let us now find b_k. The representation of $\mathcal{O}_{\Delta_K}$ is $\left(1, \frac{\Delta_K + \sqrt{\Delta_K}}{2}\right)$. A simple calculation gives

$$\alpha_k \mathcal{O}_{\Delta_K} = \alpha_k \mathbb{Z} + \left(\frac{k\Delta_K}{2} + \frac{\Delta_K(\Delta_K + 1)}{4} + (k + \Delta_K)\frac{\sqrt{\Delta_K}}{2}\right)\mathbb{Z}$$

which must be equal to $m_k\left(a_k\mathbb{Z}+\frac{-b_k+\sqrt{\Delta_K}}{2}\mathbb{Z}\right)$. As mentioned in the proof of [BTW95, Proposition 2.9], m_k is the smallest positive coefficient of $\sqrt{\Delta_K}/2$ in $\mathfrak{A}_k$: in our case $m_k=\gcd(1,k+\Delta_K)$ and therefore $m_k=1$.

Since $\alpha_k\in\alpha_k\mathcal{O}_{\Delta_K}$, there exists μ_k and ν_k such that $\alpha_k=a_k\mu_k+\frac{-b_k+\sqrt{\Delta_K}}{2}\nu_k$. By identification in the basis $(1,\sqrt{\Delta_K})$, $\nu_k=1$ and by a multiplication by 2, we obtain $2k+\Delta_K=2a_k\mu_k-b_k$. As the value of b_k is defined modulo $2a_k$, we can take

$$b_k=-2k-\Delta_K.$$

We now need to compute $\varphi_q^{-1}(\mathfrak{A}_k)$. From Algorithm 1, it is equal to $(a_k, b_kq \bmod 2a_k)$. Eventually, in every nontrivial class of $\ker\bar{\varphi}_q$, there exists an ideal (a_k,b_kq). This ideal corresponds to the quadratic form $a_kx^2+b_kqxy+c_ky^2$ with

$$c_k=\frac{q^2\left((2k+\Delta_K)^2-\Delta_K\right)}{4(k^2+\Delta_Kk)+\Delta_K(\Delta_K-1)}=q^2,$$

which is then equivalent to the form $q^2x^2-b_kqxy+a_ky^2$ corresponding to the ideal $(q^2,-b_kq)$ whose norm is q^2. □

A first attempt: inverting the reduction process. From this theorem, the reduced ideal $\mathfrak{h}$ published in the NICE cryptosystems is equivalent to an ideal of norm q^2. A first attack is thus to try to do a brute force ascent of the reduction algorithm, *i. e.*, the Gauss algorithm, from $\mathfrak{h}$. To "invert" a step of this algorithm (see Algorithms 1.3.14 and 5.4.2 of [Coh00]), one has to consider all the possible quotients of the Euclidean division. The number of possible quotients is heuristically low (say ten), and the complexity of the attack grows exponentially with the number of reduction steps. If this number is very low, the attack will be feasible. In particular, if $q<\sqrt{p/4}$, all ideals of the form (q^2,kq) are already reduced, so the norm of $\mathfrak{h}$ is q^2 and the schemes are insecure. If the parameters for NICE are chosen as proposed in [PT00] (*i. e.*, $\sqrt{p/3}<q$) the number of reduction steps can still be too low. In the given implementation and later papers (*e. g.*, [BST02]), p and q are actually chosen of same size λ, the security parameter. Let us analyse more generally the numbers of reduction steps needed to reduce ideals of the form (q^2,kq) in $C(\mathcal{O}_{\Delta_q})$.

If we translate the problem in terms of quadratic forms, the quadratic form $q^2x^2+kqxy+c(k)^2y^2$, with $c(k):=\frac{1}{4}(k^2+p)$, can be represented by the matrix

$$\begin{pmatrix} q^2 & kq/2 \\ kq/2 & c(k)\end{pmatrix},$$

which defines (up to an isometry) two vectors u and v of $\mathbb{C}$ such that $|u|^2=q^2$, $|v|^2=c(k)$ and $\langle u,v\rangle=kq/2$, where $\langle\cdot,\cdot\rangle$ denotes the usual scalar product in $\mathbb{C}$. If we consider the complex number $z=\frac{v}{u}$ (we suppose here that u is larger than v, *i. e.*, $q^2>\frac{1}{4}(k^2+p)$), then

$$z=\frac{\langle u,v\rangle}{|u|^2}+i\frac{\det(u,v)}{|u|^2}=\frac{kq}{2q^2}+i\frac{\sqrt{|\Delta_q|}}{q^2}=\frac{k}{2q}+i\frac{\sqrt{p}}{q}\cdot$$

The mean number of iteration A_h of the Gauss algorithm when the complex number z belongs to the strip $\{|\Im(z)| \leq 1/h\}$ is heuristically

$$A_h \sim \frac{1}{2}\log h \left[\frac{1}{\log(1+\sqrt{2})} - \frac{1}{\log\left(\frac{\pi^2}{6\log\phi}\right)}\right],$$

where ϕ is the golden ratio.

Inside this horizontal strip, the complex numbers z for which the number of iterations is of order $\Omega(\log L)$ are those for which their real part $\Re(z)$ is close to a rational number whose continued fraction expansion is of order $\Omega(\log L)$.

Then, since our complex number z is of the form $z = \frac{k}{2q} + i\frac{\sqrt{p}}{q}$, the number of iterations of the Gauss Algorithm on the input z will be (with a high probability) of order $\Omega(\log qp^{-\frac{1}{2}})$ provided that the height of the continued fraction expansion of the rational number k/q is of order $\Omega(\log q)$ (which is always the case, with a high probability). See [VV07] for a precise analysis of Gauss algorithm. If we set $q = p^\alpha$ these theoretical results give a behaviour in $\Omega\big(\big(\alpha - \frac{1}{2}\big)\log p\big)$, and therefore if we set $\alpha = 1$ as suggested in [BST02], we have a number of steps proportional to $\log p/2 = \lambda/2$ so the going up is infeasible. Note that our experiments confirm this complexity. Therefore we have to establish another strategy to recover these non-reduced ideal of norm q^2.

3.2 An Algorithm to Solve the Kernel Problem

Description. In this subsection, we describe an algorithm which totally breaks the NICE family of cryptosystems by solving the Kernel Problem in polynomial time in the security parameter. More precisely, given $\Delta_q = -pq^2$ where p and q are two λ-bit primes and $\mathfrak{h}$ a reduced ideal whose class is in the kernel of the surjection from $C(\mathcal{O}_{\Delta_q})$ to $C(\mathcal{O}_{\Delta_K})$, this algorithm outputs p and q in cubic time. The next subsection is dedicated to the analysis of the correctness and the complexity of this algorithm. The main result is given in Corollary 1.

The strategy of the attack, detailed in the next algorithm, is as follows. First, in an initialisation phase (steps 1–3), we generate a power r of a small odd prime. This integer r is chosen large enough to make the ideals of norm q^2 reduced in $C(\mathcal{O}_{\Delta_q r^2})$. Then, the core of the algorithm consists in lifting $[\mathfrak{h}']$ (where $\mathfrak{h}'$ is equivalent to $\mathfrak{h}$ and prime to r) in this class group. In step 5, we compute $\mathfrak{g} = \mathfrak{h}' \cap \mathcal{O}_{\Delta_q r^2}$, which is an $\mathcal{O}_{\Delta_q r^2}$-ideal, with Algorithm 1 (this algorithm still works between two non-maximal orders).

Then, in step 6, we compute the reduced element $\mathfrak{f}$ of the class of $\mathfrak{g}$ raised to the power $\phi_{\Delta_K}(r)$. In the next subsection, we will prove that this lift (steps 5 and 6) maps almost all the elements of $\ker\bar{\varphi}_q$, including $[\mathfrak{h}]$, to elements of $\ker\bar{\varphi}_{qr}$ whose reduced ideal has norm q^2. As a consequence, the ideal $\mathfrak{f}$ computed in step 6 has norm q^2 and eventually step 7 extracts p and q.

Input: $\lambda \in \mathbb{Z}, \Delta_q = -pq^2 \in \mathbb{Z}, \mathfrak{h} = (a, b) \in I(\mathcal{O}_{\Delta_q}, q)$ with $[\mathfrak{h}] \in \ker \bar{\varphi}_q$ of order > 6

Output: p, q

Initialisation:

1. Set $r' = 3$
2. Set $\delta_{r'} = \lceil \frac{\lambda+3}{2} \frac{\log 2}{\log r'} \rceil$ and $r = r'^{\delta_{r'}}$
3. **If** the order of $[\mathfrak{h}]$ divides $\phi_{\Delta_K}(r)$ **then** set r' to the next prime and **goto** 2.
4. Find $\mathfrak{h}' \in [\mathfrak{h}]$ such that $\mathfrak{h}' \in I(\mathcal{O}_{\Delta_q}, r')$ [HJPT98, Algorithm 1]

Core Algorithm:

5. Compute $\mathfrak{g} = \mathfrak{h}' \cap \mathcal{O}_{\Delta_q r^2}$ [Algorithm 1]
6. Compute $\mathfrak{f} = \mathsf{Red}(\mathfrak{g}^{\phi_{\Delta_K}(r)})$
7. **Return** $p = \Delta_q / \mathrm{N}(\mathfrak{f})$, $q = \sqrt{\mathrm{N}(\mathfrak{f})}$

Algorithm 3: Solving the Kernel Problem

Remark 2. We omit elements of small order in the input of our algorithm, because they are useless for the NICE cryptosystems. As we will see in the proof of Corollary 1, this restriction ensures that the incrementation of step 3 will be done at most once. For completeness, if the order of $[\mathfrak{h}]$ is 3, only few iterations will be done to obtain a suitable r such that the order of $[\mathfrak{h}]$ does not divide $\phi_{\Delta_K}(r) = r'^{\delta_{r'}-1}\left(r' - \left(\frac{\Delta_K}{r'}\right)\right)$, and for an order of 5, $r' = 3$ suits. Note also that elements of order 2 (4 and 6) leads to ambiguous ideals which give the factorisation of the discriminant (see [Sch82]).

Correctness. Again, the proof of correctness of Algorithm 3 will be done by using the effective isomorphisms between $\ker \bar{\varphi}_q$ and $(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times$ and between $\ker \bar{\varphi}_{qr}$ and $(\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times$. The integer r is an odd integer prime to q and Δ_K such that $r > 2q/\sqrt{|\Delta_K|}$, *i. e.*, such that ideals of norm q^2 are reduced in $C(\mathcal{O}_{\Delta_q r^2})$.

First in Lemma 2, we prove that nontrivial elements of a certain subgroup of the quotient $(\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times$ map to classes of $\ker \bar{\varphi}_{qr}$ whose reduced element has norm q^2. Actually, this subgroup contains the image of a particular lift of $(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times$ following the Chinese remainder theorem: A class $[\alpha]$ modulo q is lifted to a class $[\beta]$ modulo qr such that $[\beta] \equiv 1 \pmod{r}$ and $[\beta] \equiv [\alpha]^{\phi_{\Delta_K}(r)} \pmod{q}$.

Then, in Theorem 3, we prove that the lift computed in steps 4 and 6 of Algorithm 3 corresponds to the lift previously mentioned on the quotients of $\mathcal{O}_{\Delta_K}$. As a result, this lift evaluated on an element of $\ker \bar{\varphi}_q$ either gives the trivial class or a class corresponding to the nontrivial elements of the subgroup of Lemma 2, *i. e.*, a class whose reduced element has norm q^2.

Finally, in Corollary 1, we prove that Algorithm 3 is polynomial and correct, *i. e.*, that the choice of r done in the initialisation of the algorithm ensures that the lift will produce a nontrivial class and hence an ideal of norm q^2.

Lemma 2. *Let Δ_K be a fundamental negative discriminant, different from -3 and -4 and q an odd prime conductor and r be an odd integer prime to q and Δ_K such that $r > 2q/\sqrt{|\Delta_K|}$. The isomorphism ψ_{qr} of Lemma 1 maps the nontrivial elements of the kernel of this natural surjection*

$$\pi : (\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times \twoheadrightarrow (\mathcal{O}_{\Delta_K}/r\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/r\mathbb{Z})^\times$$

to classes of $\ker \bar{\varphi}_{qr} \subset C(\mathcal{O}_{\Delta_K q^2 r^2})$*, whose reduced element has norm q^2.*

Proof. This proof is similar to the proof of Theorem 2, but relative to r (more precisely, specialising $r = 1$ in this lemma yields Theorem 2). Let us set $\alpha_k = k + r\frac{\Delta_K + \sqrt{\Delta_K}}{2}$ where $k \in \mathbb{Z}$ takes $\phi_{\Delta_K}(q)$ values s.t. $\begin{cases} k \not\equiv 0 \pmod r, \\ k \equiv 0, \ldots, q-1 \pmod q, \\ k^2 \not\equiv r^2\Delta_K \pmod q. \end{cases}$
and denote $\mathcal{S} = \{1\} \cup \{\alpha_k\}_k$. For each k, the norm $\mathrm{N}(\alpha_k)$ is equal to $\left(k + r\frac{\Delta_K}{2}\right)^2 - \Delta_K\frac{r^2}{4}$.

Since r is prime to q, the Chinese remainder theorem gives the isomorphism between $(\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times$ and

$$\left((\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times\right) \times \left((\mathcal{O}_{\Delta_K}/r\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/r\mathbb{Z})^\times\right).$$

As all the elements of $\mathcal{S}$ map to the neutral element in $(\mathcal{O}_{\Delta_K}/r\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/r\mathbb{Z})^\times$ and gives all the elements of $(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times$, $\mathcal{S}$ is actually a set of representatives of $\ker \pi$.

Let us now compute $\mathfrak{A}_k = \alpha_k\mathcal{O}_{\Delta_K}$. Its representation is (a_k, b_k), with $a_k = \mathrm{N}(\alpha_k)$ and then

$$\alpha_k\mathcal{O}_{\Delta_K} = \alpha_k\mathbb{Z} + \left(k + r\frac{\Delta_K + \sqrt{\Delta_K}}{2}\right)\left(\frac{\Delta_K + \sqrt{\Delta_K}}{2}\right)\mathbb{Z},$$

which must be equal to $m_k\left(a_k\mathbb{Z} + \frac{-b_k + \sqrt{\Delta_K}}{2}\mathbb{Z}\right)$. The integer m_k is then equal to $\gcd(r, r\Delta_K + k)$ which is equal to 1 since $\gcd(k, r) = 1$.

As $\alpha_k \in \alpha_k\mathcal{O}_{\Delta_K}$, there exists μ_k and ν_k such that $\alpha_k = a_k\mu_k + \frac{-b_k + \sqrt{\Delta_K}}{2}\nu_k$. By identification in the basis $(1, \sqrt{\Delta_K})$, $\nu_k = r$ and by multiplying by 2, we obtain $2k + r\Delta_K = 2a_k\mu_k - rb_k$ and again we can take

$$b_k = \frac{-2k}{r} - \Delta_K.$$

Then $\varphi_{qr}^{-1}(\mathfrak{A}_k)$ is equal to (a_k, B_k) where $B_k = b_k qr$. This ideal corresponds to the quadratic form $a_k x^2 + B_k xy + c_k y^2$ with

$$c_k = \frac{B_k^2 - q^2r^2\Delta_K}{4a_k} = q^2,$$

which is then equivalent to the form $q^2x^2 - B_kxy + a_ky^2$ corresponding to the ideal $(q^2, -B_k) = (q^2, -B_k \mod_c 2q^2)$, where the subscript c designates the centered euclidean division. Finally, this ideal is reduced because $|-B_k \mod_c 2q^2| < q^2 < \sqrt{\Delta_K q^2 r^2/4}$. □

Theorem 3. *Let Δ_K be a fundamental negative discriminant, different from -3 and -4 and q be an odd prime conductor. Let r be an odd integer, prime to both q and Δ_K such that $r > 2q/\sqrt{|\Delta_K|}$. Given a class of* $\ker\bar{\varphi}_q$ *and $\mathfrak{h}$ a representative in $I(\mathcal{O}_{\Delta_q}, qr)$, then the class*

$$[\mathfrak{h} \cap \mathcal{O}_{\Delta_q r^2}]^{\phi_{\Delta_K}(r)}$$

is trivial if the order of $[\mathfrak{h}]$ divides $\phi_{\Delta_K}(r)$ and has a reduced element of norm q^2 otherwise.

Proof. Let $\mathfrak{h} \in I(\mathcal{O}_{\Delta_q}, q)$ be a representative of a class of $\ker\bar{\varphi}_q$. Let $\alpha \in \mathcal{O}_{\Delta_K}$ such that $\mathfrak{h}\mathcal{O}_{\Delta_K} = \alpha\mathcal{O}_{\Delta_K}$. Let us remark first that $\mathfrak{h} \cap \mathcal{O}_{\Delta_q r^2}$, which is an $\mathcal{O}_{\Delta_q r^2}$-ideal, is equal to $\alpha\mathcal{O}_{\Delta_K} \cap \mathcal{O}_{\Delta_q r^2}$. Therefore $[\mathfrak{h} \cap \mathcal{O}_{\Delta_q r^2}]$ is in $\ker\bar{\varphi}_{qr}$. By the isomorphisms of Lemma 1, $[\mathfrak{h}] \in \ker\bar{\varphi}_q$ corresponds to $\big([\alpha] \pmod q\big) \in (\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times$ and $[\mathfrak{h} \cap \mathcal{O}_{\Delta_q r^2}]$ corresponds to $\big([\alpha] \pmod{qr}\big)$ in the quotient $(\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times$.

Once again, we are going to use properties of quotients of $\mathcal{O}_{\Delta_K}$ to obtain some information on the kernel of $\bar{\varphi}_q$ and $\bar{\varphi}_{qr}$. Let

$$\begin{array}{ccccc} s : & (\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times & \longrightarrow & (\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times \\ & [\alpha] & \longmapsto & [\alpha]^{\phi_{\Delta_K}(r)} . \end{array}$$

The map s is a well-defined morphism. Indeed, if α and β are two elements of $\mathcal{O}_{\Delta_K}$ such that $[\alpha] = [\beta]$ in $(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times$, then, in the Chinese remainder isomorphism (describing the quotient $(\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times$), $[\alpha]^{\phi_{\Delta_K}(r)}$ maps to $([\alpha]^{\phi_{\Delta_K}(r)} \pmod q, [1] \pmod r)$. On the other hand, the element $[\beta]^{\phi_{\Delta_K}(r)}$ maps to $([\beta]^{\phi_{\Delta_K}(r)} \pmod q, [1] \pmod r)$ and therefore $s([\alpha]) = s([\beta])$. Note that the kernel of s is the subgroup of $\phi_{\Delta_K}(r)$-th roots of unity of $(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times$.

Let us define the morphism $\hat{s}$ between $\ker\bar{\varphi}_q$ and $\ker\bar{\varphi}_{qr}$ such that the following diagram commutes:

$$\begin{array}{ccc} \ker\bar{\varphi}_q & \xrightarrow{\hat{s}} & \ker\bar{\varphi}_{qr} \\ \uparrow \psi_q \wr & \circlearrowleft & \uparrow \psi_{qr} \wr \\ (\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/q\mathbb{Z})^\times & \xrightarrow{s} & (\mathcal{O}_{\Delta_K}/qr\mathcal{O}_{\Delta_K})^\times / (\mathbb{Z}/qr\mathbb{Z})^\times \end{array}$$

Now, we prove that $\hat{s}([\mathfrak{h}]) = [\mathfrak{h} \cap \mathcal{O}_{\Delta_q r^2}]^{\phi_{\Delta_K}(r)}$. Indeed, $\hat{s}([\mathfrak{h}]) = \hat{s} \circ \psi_q([\alpha] \pmod q)$ and by commutativity of the diagram

$$\begin{aligned}\hat{s} \circ \psi_q([\alpha] \pmod q) &= \psi_{qr} \circ s([\alpha] \pmod q) \\ &= \psi_{qr}\Big(([\alpha] \pmod q)^{\phi_{\Delta_K}(r)}\Big) \\ &= \psi_{qr}\Big(([\alpha] \pmod{qr})^{\phi_{\Delta_K}(r)}\Big) \\ &= \psi_{qr}\Big([\alpha] \pmod{qr}\Big)^{\phi_{\Delta_K}(r)} = [\mathfrak{h} \cap \mathcal{O}_{\Delta_q r^2}]^{\phi_{\Delta_K}(r)}.\end{aligned}$$

By construction, $\ker \hat{s}$ is the subgroup of $\phi_{\Delta_K}(r)$-th roots of unity of $\ker \bar{\varphi}_q$ and therefore, if the order of $[\mathfrak{h}]$ divides $\phi_{\Delta_K}(r)$, then $\hat{s}([\mathfrak{h}]) = [\mathcal{O}_{\Delta_q r^2}]$. Otherwise, as the image of s is a subset of the kernel of the surjection π of Lemma 2, the reduced ideal of the class $\hat{s}([\mathfrak{h}])$ has norm q^2. □

Corollary 1. *Algorithm 3 solves the Kernel Problem and totally breaks the* NICE *family of cryptosystems in cubic time in the security parameter.*

Proof. The correctness of Algorithm 3 follows from the previous theorem: All the assumptions are verified. In particular, $r > 2q/\sqrt{|\Delta_K|}$ and $\mathfrak{h}'$ is a representative of $[\mathfrak{h}]$ in $I(\mathcal{O}_{\Delta_q}, qr)$: The ideal $\mathfrak{h}'$ is chosen prime to r' and will be also prime to q, otherwise the factorisation of Δ_q is already recovered. Now, $[\mathfrak{t}]$ is trivial if the order of $[\mathfrak{h}]$ divides $\phi_{\Delta_K}(r) = r'^{\delta_{r'}-1}\left(r' - \left(\frac{\Delta_K}{r'}\right)\right)$. As we suppose that the order of $[\mathfrak{h}]$ is greater than 6 (see Remark. 2), at most one iteration of step 3 will be done, otherwise the order of $[\mathfrak{h}]$ divides both $\phi_{\Delta_K}(3^{\delta_3})$ and $\phi_{\Delta_K}(5^{\delta_5})$, which is impossible (since their gcd is 2, 4 or 6, according to the value of the Kronecker symbols). Eventually, $\mathfrak{t}$ has norm q^2 and therefore Algorithm 3 outputs a nontrivial factorisation of Δ_q.

The cost of the initialisation phase is essentially cubic in the security parameter. The core of the algorithm consists in applying Algorithm 1 whose complexity is quadratic in λ, and an exponentiation whose complexity is cubic. □

Corollary 1 implies that all the schemes for which a public element of the kernel of $\bar{\varphi}_q$ is needed are broken in polynomial time. This includes the NICE encryption scheme and its variants, notably the enhanced IND-CCA2 version (cf. [PT99, HPT99,PT00,Huh00,BST02]), the derived signature scheme (cf. [HM00,Huh01]) and the undeniable signature scheme (cf. [BPT04]). Note that this result does not affect the security of the adaptation of seminal cryptosystems in imaginary quadratic fields, *i. e.*, the Diffie-Hellman key exchange of [BW88, McC89], the Rabin and RSA analogues of [HMT99] and the adaptation of Elgamal of [HJPT98].

Example. We apply our cryptanalysis on the example of the NICE encryption scheme mentioned in [JJ00], described as follows:

$$\begin{aligned}\Delta_q = &-100113361940284675007391903708261917456537242594667\\ &491514934053946421992795516818216760083640752198709726199732701843864411853249644535365728802022498185665592983708546453282107912775914256762913490132215200\\ &22224671621236001656120923\end{aligned}$$

$$\begin{aligned}a = &\,57022687708942583181685884381175588713007831807699951\\ &95092715895755173700399141486895731384747\end{aligned}$$

$$\begin{aligned}b = &\,33612360405827547849585862980179491106487317456059301\\ &64666819569606755029773074415823039847007\end{aligned}$$

The public key consists in Δ_q and $\mathfrak{h} = (a, b)$.

The ideal $\mathfrak{h} = (a, b)$ is equivalent to the ideal $\mathfrak{h}' = (a', b')$ with norm prime to 3 with $b' = -b$ and $a' = (b^2 - \Delta_q)/4a$:

$$\begin{aligned}a' = &\,43891898980317792308326285455049173482378605867\\ &42403785190862097985269408138288879224220052968\\ &10150815323915182343893632698778887397967669\end{aligned}$$

$$\begin{aligned}b' = &-33612360405827547849585862980179491106487317456059301\\ &64666819569606755029773074415823039847007\end{aligned}$$

We used the following power of 3:

$$r = 3^{83} = 399083839418733992953424667557234903522 7$$

Then, in 20ms, we have computed the lift of (a', b') of norm q^2:

$$\begin{aligned}\mathfrak{t} = (&\,53631231719770388398296099992823384509917463282369573510894245774887056120365979002534633233830227721465513935614971593907712680952249981870640736401120729,\\ &50726115195894796350644539158073328654518399170010324260439808053865626730478159167292645232706489579615441563764090965623987919889655079184915879970067243)\end{aligned}$$

The experiments have been done on a standard laptop running Linux with PARI/GP.

4 Conclusion

We totally break a large class of cryptosystems based on imaginary quadratic field arithmetic, whose main interest was the quadratic complexity of the secret operation. This polynomial time attack shows that SKEP and the kernel problem are not suited to build cryptosystems and lessen the number of public-key

cryptosystems with quadratic decryption time. The adaptation of NICE recently proposed in [JSW08] in the very different setting of real quadratic fields, seems to resist to our attack.

Acknowledgement. We warmly thank Denis Simon with whom we had helpful discussions on quadratic forms, Brigitte Vallée for her huge contribution to the analysis of the complexity of our first attack, and last not but least Andreas Enge for his conscientious reviewing of a preliminary version of our paper and for his precious comments.

References

[BPT04] Biehl, I., Paulus, S., Takagi, T.: Efficient Undeniable Signature Schemes based on Ideal Arithmetic in Quadratic Orders. Des. Codes Cryptography 31(2), 99–123 (2004)

[BDH99] Boneh, D., Durfee, G., Howgrave-Graham, N.: Factoring $N = p^r q$ for large r. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 326–337. Springer, Heidelberg (1999)

[BST02] Buchmann, J., Sakurai, K., Takagi, T.: An IND-CCA2 Public-Key Cryptosystem with Fast Decryption. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 51–71. Springer, Heidelberg (2002)

[BW88] Buchmann, J., Williams, H.C.: A Key-Exchange System based on Imaginary Quadratic Fields. J. Cryptology 1, 107–118 (1988)

[BTW95] Buchmann, J., Thiel, C., Williams, H.C.: Short Representation of Quadratic Integers. In: Proc. of CANT 1992, Math. Appl., vol. 325, pp. 159–185. Kluwer Academic Press, Dordrecht (1995)

[CNP99] Coron, J.-S., Naccache, D., Paillier, P.: Accelerating Okamoto-Uchiyama public-key cryptosystem. Electronics Letters 35(4), 291–292 (1999)

[Coh00] Cohen, H.: A Course in Computational Algebraic Number Theory. Springer, Heidelberg (2000)

[Cox99] Cox, D.A.: Primes of the form $x^2 + ny^2$. John Wiley & Sons, Chichester (1999)

[HM89] Hafner, J.L., McCurley, K.S.: A Rigorous Subexponential Algorithm for Computation of Class Group. J. Amer. Math. Soc. 2(4), 837–850 (1989)

[HPT99] Hartmann, M., Paulus, S., Takagi, T.: NICE - New Ideal Coset Encryption. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 328–339. Springer, Heidelberg (1999)

[Huh00] Hühnlein, D.: Efficient Implementation of Cryptosystems Based on Non-maximal Imaginary Quadratic Orders. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 147–167. Springer, Heidelberg (2000)

[Huh01] Hühnlein, D.: Faster Generation of NICE-Schnorr-Type Signatures. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 1–12. Springer, Heidelberg (2001)

[HJPT98] Hühnlein, D., Jacobson Jr., M.J., Paulus, S., Takagi, T.: A Cryptosystem Based on Non-maximal Imaginary Quadratic Orders with Fast Decryption. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 294–307. Springer, Heidelberg (1998)

[HJW03] Hühnlein, D., Jacobson Jr., M., Weber, D.: Towards Practical Non Interactive Public-Key Cryptosystems Using Non-Maximal Imaginary Quadratic Orders. Des. Codes Cryptography 30(3), 281–299 (2003)

[HM00] Hühnlein, D., Merkle, J.: An Efficient NICE-Schnorr-Type Signature Scheme. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 14–27. Springer, Heidelberg (2000)

[HMT99] Hühnlein, D., Meyer, A., Takagi, T.: Rabin and RSA Analogues Based on Non-maximal Imaginary Quadratic Orders. In: Proc. of ICISC 1998, pp. 221–240 (1999)

[JSW08] Jacobson Jr., M.J., Scheidler, R., Weimer, D.: An Adaptation of the NICE Cryptosystem to Real Quadratic Orders. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 191–208. Springer, Heidelberg (2008)

[JJ00] Jaulmes, É., Joux, A.: A NICE cryptanalysis. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 382–391. Springer, Heidelberg (2000)

[LL93] Lenstra, A.K., Lenstra Jr., H.W. (eds.): AMCP 1998. LNM, vol. 1554, p. 131. Springer, Heidelberg (1993)

[Len87] Lenstra Jr., H.W.: Factoring integers with elliptic curves. Annals of Mathematics 126(2), 649–673 (1987)

[McC89] McCurley, K.S.: Cryptographic Key Distribution and Computation in Class Groups. In: Proc. of NATO ASI on Number Theory and Applications, pp. 459–479. Kluwer Academic Press, Dordrecht (1989)

[OP01] Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)

[Pai99] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)

[Poi00] Pointcheval, D.: Chosen-Ciphertext Security for Any One-Way Cryptosystem. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 129–146. Springer, Heidelberg (2000)

[Poi05] Pointcheval, D.: Provable Security for Public Key Schemes. In: Advanced Courses CRM Barcelona, Advanced Course on Contemporary Cryptology, pp. 133–189. Birkhäuser Publishers, Basel (2005)

[PT99] Paulus, S., Takagi, T.: A generalization of the Diffie-Hellman problem and related cryptosystems allowing fast decryption. In: Proc. of ICISC 1998, pp. 211–220 (1999)

[PT00] Paulus, S., Takagi, T.: A New Public-Key Cryptosystem over a Quadratic Order with Quadratic Decryption Time. J. Cryptology 13(2), 263–272 (2000)

[Sch82] Schoof, R.: Quadratic fields and factorization. Computational Methods in Number Theory, MC-Tracts 154/155, 235–286 (1982)

[Sch00] Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)

[VV07] Vallée, B., Vera, A.: Lattice Reduction in Two Dimensions: Analyses under Realistic Probabilistic Models. In: Proc. of AofA 2007, DMTCS. AH, pp. 181–216 (2007)

Cube Attacks on Tweakable Black Box Polynomials

Itai Dinur and Adi Shamir

Computer Science department
The Weizmann Institute
Rehobot 76100, Israel

Abstract. Almost any cryptographic scheme can be described by *tweakable polynomials* over $GF(2)$, which contain both secret variables (e.g., key bits) and public variables (e.g., plaintext bits or IV bits). The cryptanalyst is allowed to tweak the polynomials by choosing arbitrary values for the public variables, and his goal is to solve the resultant system of polynomial equations in terms of their common secret variables. In this paper we develop a new technique (called a *cube attack*) for solving such tweakable polynomials, which is a major improvement over several previously published attacks of the same type. For example, on the stream cipher Trivium with a reduced number of initialization rounds, the best previous attack (due to Fischer, Khazaei, and Meier) requires a barely practical complexity of 2^{55} to attack 672 initialization rounds, whereas a cube attack can find the complete key of the same variant in 2^{19} bit operations (which take less than a second on a single PC). Trivium with 735 initialization rounds (which could not be attacked by any previous technique) can now be broken with 2^{30} bit operations. Trivium with 767 initialization rounds can now be broken with 2^{45} bit operations, and the complexity of the attack can almost certainly be further reduced to about 2^{36} bit operations. Whereas previous attacks were heuristic, had to be adapted to each cryptosystem, had no general complexity bounds, and were not expected to succeed on random looking polynomials, cube attacks are provably successful when applied to random polynomials of degree d over n secret variables whenever the number m of public variables exceeds $d + log_d n$. Their complexity is $2^{d-1}n + n^2$ bit operations, which is polynomial in n and amazingly low when d is small. Cube attacks can be applied to any block cipher, stream cipher, or MAC which is provided as a black box (even when nothing is known about its internal structure) as long as at least one output bit can be represented by (an unknown) polynomial of relatively low degree in the secret and public variables.

Keywords: Cryptanalysis, algebraic attacks, cube attacks, tweakable black box polynomials, stream ciphers, Trivium.

1 Introduction

Solving large systems of multivariate polynomial equations is considered an exceedingly difficult problem, which had been studied extensively over many years.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 278–299, 2009.

The problem is NP-complete even when the system contains only quadratic equations modulo 2 (see [18]), and it provides the main protective mechanism in many cryptographic schemes.

The main mathematical tool developed in order to solve such equations is the notion of Grobner bases (see [1],[2] and [3]), but when we try to apply it in practice to random equations with more than 100 variables it usually runs out of space without providing any answers. The much simpler linearization technique considers each term in these polynomials as a new independent variable, and tries to solve the resultant system of linear equations by Gauss elimination. Its main problem is that it requires a hugely overdefined system of polynomial equations. For example, a system of 256 polynomial equations of degree $d = 16$ in $n = 256$ variables over $GF(2)$ is expected to have a unique solution, but in order to find it by linearization we have to increase the number of equations to the number of possible terms in these equations, which is about $n^d = 2^{128}$. There are several improved algorithms such as XL and XSL (see [3],[4],[5], [6] and [7]) which reduce the number of required equations and the time and space complexities, but they are still completely impractical for such sizes.

The main observation in this paper is that the polynomial equations defined by many cryptographic schemes are not arbitrary and unrelated. Instead, they are typically variants derived from a single *master polynomial* by setting some *tweakable variables* to any desired value by the attacker. For example, in block ciphers and message authentication codes (MAC's) the output depends on key bits which are secret and fixed, and on message bits which are public and controllable by the attacker in a chosen plaintext attack. Similarly, in stream ciphers the output depends on secret fixed key bits and on public IV bits which can be chosen arbitrarily. By modifying the values of these tweakable public bits, the attacker can obtain many *derived polynomial equations* which are closely related. What we show in this paper is that when the master polynomial is sufficiently random, we can eliminate with *provably high probability* all of its n^d nonlinear terms by considering a surprisingly small number of only $2^d n$ tweaked variants, and then solve a precomputed version of the resultant n linear equations in n variables using only n^2 bit operations. For example, when $d = 16$ and $n = 10,000$, we can simultaneously eliminate all the 2^{200} nonlinear terms by considering only the 2^{20} derived polynomial equations obtained by encrypting 2^{20} chosen plaintexts defined by setting 20 public bits to all their possible values. After this "massacre" of nonlinear terms, the only thing left is a random looking system of linear equations in all the secret variables, which is easy to solve. In case the master polynomial is not random, there are no guarantees about the success rate of the attack, and if the degree of the master polynomial is too high, the basic attack technique is not likely to work. For these cases, we describe in the appendix several generalizations which may prove to be useful.

To demonstrate the attack, consider the following dense master polynomial of degree $d = 3$ over three secret variables x_1, x_2, x_3 and three public variables v_1, v_2, v_3:

$$P(v_1, v_2, v_3, x_1, x_2, x_3) = v_1v_2v_3 + v_1v_2x_1 + v_1v_3x_1 + v_2v_3x_1 + v_1v_2x_3 + v_1v_3x_2 + v_2v_3x_2 + v_1v_3x_3 + v_1x_1x_3 + v_3x_2x_3 + x_1x_2x_3 + v_1v_2 + v_1x_3 + v_3x_1 + x_1x_2 + x_2x_3 + x_2 + v_1 + v_3 + 1$$

Third degree polynomials over six variables can have $\binom{6}{3} + \binom{6}{2} + \binom{6}{1} + \binom{6}{0} = 42$ possible terms, and thus there are 2^{42} such polynomials over $GF(2)$. To eliminate all the 35 possible nonlinear terms by Gauss elimination, we typically need 35 such polynomials. By setting the three public variables v_1, v_2, v_3 to all their possible 0/1 values, we can get only 8 derived polynomials, which seem to be insufficient. However, summing the 4 derived polynomials with $v_1 = 0$ we get $x_1 + x_2$, summing the 4 derived polynomials with $v_2 = 0$ we get $x_1 + x_2 + x_3$, and summing the four derived polynomials with $v_3 = 0$ we get $x_1 + x_3$, which simultaneously eliminated all the nonlinear terms. When we numerically sum modulo 2 the values of the derived polynomials in these three different ways (instead of symbolically summing the polynomials themselves), we get a simple system of three linear equations in the three secret variables. Consequently, the master nonlinear polynomial can be solved by a chosen message attack which evaluates it for just 8 combinations of values of its public variables.

Since we deal with dense multivariate polynomials of relatively high degree, their explicit representations are extremely big, and thus we assume that they are provided only implicitly as black boxes which can be queried. This is a natural assumption in cryptanalysis, in which the attacker can interact with an encryption black box that contains the secret key. A surprising consequence of our approach is that we can now attack completely unknown cryptosystems (such as the CRYPTO-1 algorithm implemented in millions of transportation smart cards, whose design was kept as a trade secret until very recently) which are embedded in tamper resistant hardware, without going through the tedious and expensive process of physical reverse engineering! Since the number of queries we use is much smaller than the number needed in order to uniquely interpolate the polynomial from its black box representation, our algorithm manages to break such unknown cryptosystems even when it is information theoretically impossible to uniquely determine them from the available data.

Some of the issues we deal with in this paper are how to efficiently estimate the degree d of a given black box multivariate polynomial, how to solve high degree polynomials which can be well approximated by low degree polynomials (e.g., when they only contain a small number of high degree terms which almost always evaluate to zero), and how to easily find the linear equations defined by the sums of these huge derived polynomials. Note that in the black box model the attacker is not allowed to perform symbolic operations such as asking for the coefficient of a particular term, evaluating the GCD of two polynomials, or computing their Grobner basis, unless he first interpolates them from their values by a very expensive procedure which requires a huge number of queries.

We call this cryptanalytic technique a *cube attack* since it sets some public variables to all their possible values in n (not necessarily disjoint) $(d-1)$-dimensional boolean cubes, and sums the results in each cube. The attack is not completely new, since some of its ideas and techniques were also used in previous

heuristic attacks on various cryptosystems, but we believe that this is the first time that all these elements were brought together, accompanied by careful analysis of their complexity and success rate for random black box polynomials.

Cube attacks should not be confused with the *interpolation attacks* of Jakobsen and Knudsen ([17]), which deal with cryptosystems whose basic operations are quadratic polynomials over all or half of the input. Such polynomials are univariate or bivariate polynomials over $GF(2^n)$, and thus have fairly compact representations which can be easily interpolated from sufficiently many input/output pairs. Our attack deals with huge black box multivariate polynomials over $GF(2)$ which cannot possibly be interpolated from the available data.

The attack is remotely related to the *square attack* (see [8]) which considers the special case of cryptographic schemes whose secret bits are grouped into longer words, which are arranged in a two dimensional square. Cube attacks make no such assumptions about how the secret bits in the polynomial equations are related to each other, and thus they can be applied in a much broader set of circumstances.

The attack is also superficially similar to *integral attack* (also called *saturation attack* in the literature) and to *high order differential* attack which sum the output of cryptosystems over various subsets of input variables. However, as explained in section 3, this is just an artifact of the special field $GF(2)$ in which addition and subtraction are the same operation, and over a general field $GF(p^k)$ with $p > 2$ we have to use a different way to apply cube attacks.

Several previously published techniques try to break particular schemes by highly heuristic attacks that sum output values on some Boolean cubes of public variables. These related attacks include [26], [27], [28], [29], [30] and [31], and are collectively referred to as *chosen IV statistical attacks*. Compared to these attacks, the cube attack is much more general, is applicable to block ciphers in addition to stream ciphers, and has a better-defined preprocessing phase which does not need adaptations for each given scheme. As a result, cube attacks can be applied with provable success rate and complexity even when the cryptosystem is modelled by a random black box polynomial about which nothing is known. The most important difference is that in cube attacks each summation leads to an easily solvable linear equation (in any number of secret key bits), whereas in chosen IV statistical techniques there are many attack scenarios, and each summation typically leads only to a statistically biased expression (in a small subset of the secret key bits). Such a bias has to be amplified by many repetitions using a much larger amount of data before it can be used in order to find the key. The most convincing demonstration of this difference is the best previously known chosen IV attack on the Trivium stream cipher [28]: When the number of initialization rounds is reduced to 672, this attack has a relatively high complexity of 2^{55} operations, whereas the standard unoptimized cube attack can perform full key recovery in just 2^{19} bit operations; When the number of initialization steps is increased to 735, no previously published attack is faster than exhaustive search, whereas the same cube attack can easily perform full key recovery in 2^{30} bit operations. These and further results about Trivium are discussed in the appendix.

2 Terminology

This section describes the formal notation we use in the rest of the paper. The attacker is given a black box that evaluates an unknown polynomial p over $GF(2)$ of $n+m$ inputs bits $(x_1,..,x_n,v_1,..,v_m)$ and outputs a single bit. The polynomial is assumed to be in Algebraic Normal Form, namely, the sum of products of variables. The input bits $x_1,..,x_n$ are the secret variables, while $v_1,..,v_m$ are the public variables. The solution consists of two phases. During the preprocessing phase, the attacker is allowed to set the values of all the variables $(x_1,..,x_n,v_1,..,v_m)$ and to use the black box in order to evaluate the corresponding output bit of p. This corresponds to the usual cryptanalytic setting in which the attacker can study the cryptosystem by running it with various keys and plaintexts. During the online phase, the n secret variables are set to unknown values, and the attacker is allowed to set the values of the m public variables $(v_1,..,v_m)$ to any desired values and to evaluate p on the combined input.

To simplify our notation, we ignore in the rest of this section the distinction between secret and public variables, and denote all of them by $x_1,...,x_n$. Since $x_i^2 = x_i$ modulo 2, the terms t_I in the polynomial can be indexed by the subset $I \subseteq \{1,...,n\}$ of the variables which are multiplied together, and every polynomial can be represented by sums of t_I for a certain collection of subsets I. We denote by $\mathbb{P}_d^n$ the set of all the multivariate polynomials over $GF(2)$ with n variables and total degree bounded by d.

Given a multivariate polynomial p and any index subset I, we can factor the common subterm t_I out of some of the terms in p, and represent the polynomial as the sum of terms which are supersets of I and terms which are not supersets of I:

$$p(x_1,..,x_n) \equiv t_I \cdot p_{S(I)} + q(x_1,..,x_n)$$

We call $p_{S(I)}$ the *superpoly* of I in p. Note that for any p and I, the superpoly of I in p is a polynomial that does not contain any common variable with t_I, and each term in $q(x_1,..,x_n)$ misses at least one variable from I.

To demonstrate these notions, let

$$p(x_1,x_2,x_3,x_4,x_5) = x_1x_2x_3 + x_1x_2x_4 + x_2x_4x_5 + x_1x_2 + x_2 + x_3x_5 + x_5 + 1$$

be a polynomial of degree 3 in 5 variables, and let $I = \{1,2\}$ be an index subset of size 2. We can represent p as:

$$p(x_1,x_2,x_3,x_4,x_5) = x_1x_2(x_3 + x_4 + 1) + (x_2x_4x_5 + x_3x_5 + x_2 + x_5 + 1)$$

where

$$\begin{aligned} t_I &= x_1x_2 \\ p_{S(I)} &= x_3 + x_4 + 1 \\ q(x_1,x_2,x_3,x_4,x_5) &= x_2x_4x_5 + x_3x_5 + x_2 + x_5 + 1 \end{aligned}$$

Definition 1. *A* maxterm *of p is a term t_I such that $deg(p_{S(I)}) \equiv 1$, i.e. the superpoly of I in p is a linear polynomial which is not a constant.*

Any subset I of size k defines a k-dimensional Boolean cube of 2^k vectors C_I in which we assign all the possible combinations of 0/1 values to variables in I, and leave all the other variables undetermined. Any vector $\boldsymbol{v} \in C_I$ defines a new derived polynomial $p_{|\boldsymbol{v}}$ with $n-k$ variables (whose degree may be the same or lower than the degree of the original polynomial). Summing these derived polynomials over all the 2^k possible vectors in C_I, we end up with a new polynomial, which is denoted by $p_I \triangleq \sum_{\boldsymbol{v} \in C_I} p_{|\boldsymbol{v}}$. In the next section, we prove that this polynomial has a simple alternative definition, which makes it extremely useful in cryptanalytic applications.

3 The Main Observation

Theorem 1. *For any polynomial p and subset of variables I, $p_I \equiv p_{S(I)}$ modulo 2.*

Proof. Write $p(x_1,..,x_n) \equiv t_I \cdot p_{S(I)} + q(x_1,..,x_n)$. We first examine an arbitrary term t_J of $q(x_1,..,x_n)$, where J is the subset containing the variable indexes that are multiplied together in t_J. Since t_J misses at least one of the variables in I, it is added an even number of times (for the two possible values of any one of the missed variables, where all the other values of the variables are kept the same), which cancels it out modulo 2 in $\sum_{\boldsymbol{v} \in C} p_{|\boldsymbol{v}}$.

Next, we examine the polynomial $t_I \cdot p_{S(I)}$: All $\boldsymbol{v} \in C_I$ zero t_I, except when we assign the value 1 to all the variables in I. This implies that the polynomial $p_{S(I)}$ (which has no variables with indexes in I and is thus independent of the values we sum over) is summed only once, when t_I is set to 1. Consequently, the formal sum of all the derived polynomials is exactly the superpoly $p_{S(I)}$ of the term we sum over. □

Basically, the theorem states that the sum of the 2^k polynomials derived from the original polynomial p by assigning all the possible values to the k variables in I, eliminates all the terms except those which are contained in the superpoly of I in p. The summation thus reduces the total degree of the master polynomial by at least k, and if t_I is any maxterm in p, this sum yields a linear equation in the remaining variables. For example, if we sum the polynomial $p(x_1,x_2,x_3,x_4,x_5)$ defined in the previous section over the four possible values of x_1 and x_2 in the maxterm $t_I = x_1x_2$, we get the linear expression $p_{S(I)} = (x_3 + x_4 + 1)$. Consequently, all the cryptanalyst has to do in order to solve a tweakable master polynomial of degree d is to find sufficiently many maxterms in it, and for each maxterm to sum at most 2^{d-1} derived polynomials. Note that he only has to add the 0/1 *values* of these derived polynomials (which he can obtain via a chosen plaintext attack), and not their huge symbolic expressions. The summed bit is then equated with a fixed linear expression which can be derived from the master black box polynomial during a separate preprocessing stage, since it is not key-dependent. For low degrees such as $d = 16$, the derivation of the right hand side of each linear equation during the online phase of the attack requires at most $2^{15} = 32768$ additions of single bit values, which takes a negligible amount of time.

Over a general field $GF(p^k)$ with $p > 2$, the correct way to apply cube attacks is to alternately add and subtract the outputs of the master polynomial with public inputs that range only over the two values 0 and 1 (and not over all their possible values of $0, 1, 2, ..., p-1$), where the sign is determined by the sum (modulo 2) of the vector of assigned values. In this form, they are reminiscent of FFT computations. Cube attacks are thus more closely related to high order differential attacks than to integral attacks, but they do not use the same formal operator. For example, consider the bivariate polynomial $p(x, v) = 4x^2v^3 + 3x^2v^5$ (mod 7) of degree 7. The formal derivative of this polynomial with respect to v is the 6-degree polynomial $p'_v(x, v) = 5x^2v^2 + x^2v^4$ (mod 7) whereas our numeric difference yields $p(x, 1) - p(x, 0) = (4x^2 + 3x^2) - (0 + 0) = 0$ (mod 7) which has degree 0. In addition, cube attacks use algebraic rather than statistical techniques to actually find the secret key.

4 The Preprocessing Phase

Given an explicit description of the master polynomial, it is easy to split it into $p(x_1, .., x_n) \equiv t_I \cdot p_{S(I)} + q(x_1, .., x_n)$ for any term t_I. However, when the exponentially long master polynomial is given only as a black box, it is not clear how to find this representation, and how to store it in a compact way.

When t_I is a maxterm, the issue of compact representation becomes easy, since we only have to know its superpoly $p_{S(I)}$ in order to apply the attack, and this expression is a short linear combination of some of the secret variables x_i, with the possible addition of the constant 1. Note that we can eliminate all the public variables v_i that are not summed over from this linear expression by fixing each one of them to 0 (or to 1) during the summation.

In order to actually find $p_{S(I)}$ for a given black box master polynomial and a maxterm t_I in it, we use a separate preprocessing phase in which the attacker is given the extra power of tweaking both the public and the secret variables:

Theorem 2. *Let t_I be a maxterm in a black box polynomial p. Then:*

1. *The free term in $p_{S(I)}$ can be computed by summing modulo 2 the values of p over all the inputs of $n + m$ variables which are zero everywhere except on the $d - 1$ variables in the summation cube C_I.*
2. *The coefficient of x_j in the linear expression $p_{S(I)}$ can be computed by summing modulo 2 all the values of p for input vectors which are zero everywhere except on the summation cube C_I and all the values of p for input vectors which are zero everywhere except on the summation cube and at x_j which is set to 1.*

The proof is based on the observation that in a linear expression, the coefficient of any variable x_j is 1 if and only if flipping the value of x_j flips the value of the expression, and the free term can be computed by setting all the variables to zero.

In the rest of this section, we distinguish between the cases of random and non-random master polynomials.

4.1 Preprocessing Random Polynomials

In many cryptographic schemes, the mixing of the inputs is so thorough that the representation of each ciphertext bit as a fully expanded polynomial function of the n key bits and m plaintext bits can be viewed as a random polynomial:

Definition 2. *A* random polynomial *of degree d in $n+m$ variables is a polynomial $p \in \mathbb{P}_d^{n+m}$ such that each possible term of degree at most d is independently chosen to occur with probability* 0.5*.*

In fact, the notion of randomness we need in order to lower bound the success probability of cube attacks is considerably weaker, since the only terms which play any role in the attack are those that correspond to maxterms in p:

Definition 3. *A* d-random *polynomial with $n+m$ variables is a polynomial $p \in \mathbb{P}_d^{n+m}$ such that each possible term of degree d which contains one secret variable and $d-1$ public variables is independently chosen to occur with probability* 0.5*, and all the other terms can be chosen arbitrarily.*

In any d-random polynomial, any term t_I which is the product of $d-1$ public variables v_i has an extremely high probability to be a maxterm: Its corresponding superpoly is a polynomial of degree at most 1, and it is a polynomial of degree 0 only when for all the secret variables x_i the terms $t_I x_i$ are not chosen to appear in the polynomial. The probability of this event is 2^{-n}.

For any two terms t_{I_1} and t_{I_2} which are the products of $d-1$ public variables, we get independent random choices of their corresponding superpolys, even when I_1 and I_2 are almost identical. For example, when $d = 4$, $I_1 = \{1, 2, 3\}$, and $I_2 = \{1, 2, 4\}$, each one of the two terms $v_1v_2v_3x_5$ and $v_1v_2v_4x_5$ occurs in p with probability 0.5 independently of the other. Since we do not need disjoint subsets of public variables as our maxterms, we only need about $d + \log_d n$ tweakable public variables in order to pack n different maxterms among their products, since $\binom{d+\log_d n}{d} = \binom{d+\log_d n}{\log_d n} \approx d^{\log_d n} = n$. In particular, when $d = 16$ and $n = 10,000$, it suffices to have only $m = 20$ tweakable public variables to apply the cube attack, since $\binom{20}{15} = 15,504 > n$. Note that the *computations* of these maxterms are not independent since we reuse the same derived polynomials in many overlapping cube summations, but the *results* of the computations are independent linear combinations of the secret variables.

After choosing n random maxterms, the attacker defines an $n \times n$ matrix A whose rows contain their corresponding superpolys. If the matrix is nonsingular, the attacker precomputes and stores A^{-1} in order to reduce the complexity of the linear algebra in the online phase of the attack from $O(n^3)$ to $O(n^2)$.

Since A is a random matrix in which each entry is independently selected with probability 1/2, it is very easy to compute the probability that it is nonsingular:

Lemma 1. *The probability that an $n \times n$ random binary matrix over $GF(2)$ is invertible is $\prod_{i=1}^{n}(1 - 2^{-i}) \approx 0.28879$*

Proof. The proof is by a simple induction on the rows of the matrix.

This is a constant probability, which can be made arbitrarily close to 1 during the preprocessing phase by considering a few extra maxterms. For $d = 16$ $n = 10,000$ and $m = 20$, there are $15,504$ possible superpolys to choose from, and the probability that the rank of all these random linear expressions will be smaller than $10,000$ is negligible.

Since the preprocessing phase has to be executed only once for each cryptosystem whereas the online phase has to be executed once for each key, some cryptanalytic attacks "cheat" by allowing extremely expensive operations during an unbounded preprocessing phase which make the whole attack impractical. When cube attacks are applied to random polynomials, the complexity of the preprocessing phase is at most n times larger than that of the online phase of the attack, and thus if one phase is practically feasible so is the other.

4.2 Preprocessing Nonrandom Polynomials

When the polynomial representation of the cryptosystem is not assumed to be d-random, there are no guarantees about the success rate of the attack. The basic questions we are faced with in this case are how to estimate the degree d of the polynomial p which is only given as a black box, and how to choose appropriate maxterms if they exist. We propose the following technique, which is a variant of the random walk proposed in [28].

The attacker randomly chooses a size k between 1 and m and a subset I of k public variables, and computes the value of the superpoly of I by numerically summing over the cube C_I (setting each one of the other public variables to a static value, usually to zero). If his subset I is too large, the sum will be a constant value (regardless of the choice of secret variables), and in this case he has to drop one of the public variables from I and repeat the process. If his subset I is too small, the corresponding $p_{S(I)}$ is likely to be a nonlinear function in the secret variables, and in this case he has to add a public variable to I and repeat the process. The correct choice of I is the borderline between these cases, and if it does not exist the attacker can restart with a different initial I.

The best way to understand this process is to think about a (not necessarily random) polynomial p in which all the terms have the same degree d, but contain different proportions of secret and public variables. When we sum over subsets I with $d-2$ public variables, we will get a purely quadratic polynomial in the secret variables which corresponds to all those terms that contain the $d-2$ variables in I as their public variables and two additional secret variables. Linear terms will not occur in this polynomial since every term which contains $d-1$ public variables is eliminated by at least one public variable which is not in I and is thus set to zero. Note that for nonrandom polynomials, this quadratic expression may be empty for some I (misleading us to believe that I is too large), but nonempty for another I (indicating correctly that it is too small), and thus we may have to restart the preprocessing with several initial I's. When we sum over subsets I with $d-1$ public variables, we will get a linear polynomial in the secret variables, but again it may be empty. In particular, if all the terms in the nonrandom p contain at least two secret variables, we will never be able to get any linear

superpoly during the preprocessing phase, regardless of the choice of I. When we sum over I with d public variables, we will get a key-independent constant, which is zero or one depending on whether the unique term which is the product of all the public variables in I does or does not occur in p. In this case we will always act correctly by reducing the size of I. Finally, when we sum over an I of size $d+1$ or larger, we will always get the zero polynomial, since every term in p misses at least one of the public variables in I, and will thus be added an even number of times modulo 2.

For any choice of values for all the secret variables, we sum the 0/1 values of p over the subcube C_I of public variables, setting all the other public variables to zero. This sum is a function of secret variables only, and we can test it for linearity during the preprocessing phase (in which we are allowed to modify the secret variables) by using any one of the efficient linearity tests which were developed as part of the PCP theorem (see [9]).

One example of such a linearity test is the BLR test (see [10]), which chooses vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ independently and uniformly at random, and verifies that $p_{S(I)}[\mathbf{0}] + p_{S(I)}[\boldsymbol{x}] + p_{S(I)}[\boldsymbol{y}] = p_{S(I)}[\boldsymbol{x}+\boldsymbol{y}]$. The test ensures that if $p_{S(I)}$ is linear, the test always succeeds, whereas if $p_{S(I)}$ is far from being linear, the test fails with high probability. The test is repeated sufficiently many times until the attacker is convinced that $p_{S(I)}$ is very close to being linear (e.g., it it linear, except for a few high degree terms which almost always evaluate to zero). By using the cube attack in this case, we can find most but not all of the possible keys, which is good enough in our cryptanalytic application. Note that in our preprocessing, almost all the functions we test are likely to be nonlinear superpolys (which typically fail in one of the first few linearity tests, thus requiring only a few cube summations) or easily detected constant functions, whereas in the preprocessing done by Fischer Khazaei and Meier, almost all the functions they test are balanced, and distinguishing them from slightly biased functions requires a huge number of cube summations on average.

As in the random setting, the attacker stops when sufficiently many linearly independent vectors are derived and A^{-1} can be computed. The online phase of the attack is identical to the case of random polynomials.

There are many possible optimizations of this process. For example, summing the values of p over subcubes with large intersections can be sped up by memorizing various partial sums, and thus we do not have to start from scratch when we add or eliminate one public variable from I in our proposed random walk search technique. Another extension uses the freedom to choose the values of the public variables that are not summed over. In case we get an empty superpoly for a specific cube, and a non-linear superpoly for any of its sub-cubes, we can still try to make the superpoly nonempty in order to get a maxterm by setting some of the remaining public variables to one. If the result is still zero, we can set some more of these variables to one. If the result is non-linear, we can set a few public variables that are not summed over back to zero. Note that this random walk over the *values* of the public variables we do not sum over is different from

the previously described random walk over the *subset* of the public variables we sum over.

A different attack scenario on non random polynomials uses the cube attack as a distinguisher rather than as a key extraction procedure. For example, if some output bit is a polynomial of degree at most d in the $n + m$ input variables, summing it over any d-dimensional cube of public variables will always give a constant value (which depends on the summation set I, but not on the key, and thus can be precomputed in the preprocessing phase), whereas in a random cipher such a sum will be uniformly distributed. Since the attacker has to sum over a single cube and does not have to solve any equations, the complexity of this distinguisher is just 2^d. Consequently, ANY cryptographic scheme in which $d < n$ and $d < m$ can be distinguished from a random cipher by an algorithm which is faster than exhaustive search, regardless of whether its polynomial representation is random or not. A detailed description of the theory and applications of cube distinguishers appearers in [19].

5 Applications to Block Ciphers

In chosen plaintext attacks on block ciphers, the public variables are the bits of the plaintext. Since most block ciphers have a block size of at least 128 bits, there is no shortage of tweakable variables.

Since the attack is using only a single bit from the ciphertext, it makes no difference whether the cryptographic mapping is invertible or not. Consequently, we can attack a keyed hash function (also known as a MAC, or message authentication code) by using exactly the same techniques. An example of such an attack on the keyed hash function MD6 can be found in [19].

The main problem in applying the cube attack to block ciphers is that they usually contain many rounds, and the degree of the polynomial grows exponentially with the number of rounds (until it hits the maximum possible value of $n+m$). Several techniques that may help to overcome the problem of high degree polynomials in block ciphers appear in the appendix.

6 Applications to Stream Ciphers

In the case of stream ciphers, the secret variables represent the key, and the public variables represent the IV. The model assumes that the attacker can simulate the cipher during the preprocessing phase, and can apply a chosen IV attack during the online phase. Note that we can also use a known IV attack if the stream cipher operates in the common *c*ounter mode that uses consecutive binary numbers (such as the packet number or the time of day) as its IV's, since their least significant bits contain full subcubes of various dimensions.

Many proposed stream ciphers use one or more linear feedback shift registers (LFSR), which are either filtered or combined by nonlinear functions to produce the output. In this case, the degree of the output polynomial is only determined by this function, is relatively small, is easy to bound, and does not increase when

the cipher generates a large number of bits (many of which are kept hidden during the initialization phase). The attack requires the knowledge of only one output bit for several IV values, and we can choose its location arbitrarily. In particular, we can choose a bit location in which the corresponding plaintext bit is known. Typical examples of such locations include standard packet header bits, or the high bits of ASCII characters which are known to be zero.

As an extreme example of the power of cube attacks, consider a long LFSR with $10,000$ bits and a secret dense feedback polynomial, which is filtered by a layer of $1,000$ S-boxes. Each S-box is a different secret mapping of 8 bits from the LFSR into one output bit, and the connection pattern between the LFSR and the S-boxes is also assumed to be secret. In each clock cycle, the cipher outputs only one bit, which is the XOR of the outputs of all the S-boxes. Each bit in the LFSR is initialized by a different secret dense quadratic polynomial in $10,000$ key and IV bits. The LFSR is clocked a large and secret number of times without producing any outputs, and then only the first output bit for any given IV is made available to the attacker.

The attack is a *structural attack* which is based only on the general form of the cryptosystem (as described in figure 1). Note that the attacker does not know the secret LFSR feedback polynomial, the $1,000$ S-boxes, the LFSR/S-Box interconnection pattern, the actual key/IV mixing function, or the number of dummy initialization steps. The only properties of this design which are exploited by the cube attack are that the output of each S-box is a random looking polynomial of degree 16 (obtained by substituting quadratic expressions in each one of its 8 input variables), that the XOR of these S-boxes is also a polynomial of degree 16 (in the $10,000$ secret and public variables), and that we have sufficient tweaking power over the generation of the first output bit. The attack uses only 2^{20} output bits (one for each IV value), which are summed in $10,000$ overlapping 15 dimensional cubes (note that $\binom{20}{15} = 15504 > 10000$). The attacker can thus get $10,000$ linear equations in $10,000$ variables, which he can easily solve by using the precomputed inverse of the coefficient matrix. This stream cipher can thus be broken in less than 2^{30} bit operations, even though it could not be attacked by any previous technique, including correlation attacks or the analysis of low Hamming weight LFSR modifications (see for instance [11],[12],[13],[14],[15], and [16]).

We have experimentally tested the cube attack on this stream cipher, in order to rule out the possibility that the black box polynomials which represent this stream cipher have some unexpected properties that foil the attack. In all our tests, the attack behaved exactly as expected under the assumption that the polynomials are d-random.

Some stream ciphers such as LILI and A5/1 use clock control in order to foil correlation attacks. If A5/1 had used its clock control only when producing the output bits (but not during the initialization rounds), it would have been trivial to break it with a straightforward cube attack, which uses only the first output bit produced for each IV value.

Other types of stream ciphers such as Trivium (see [21]) include a small amount of nonlinearity in the feedback of the shift register, and thus the degree

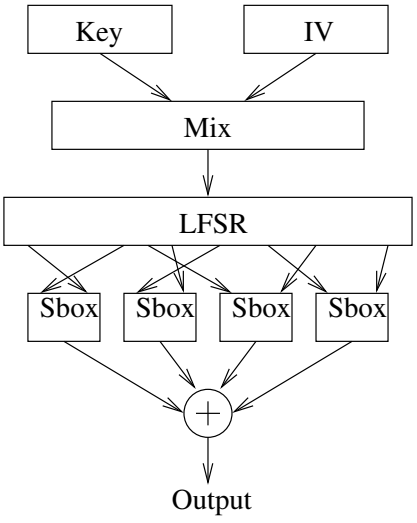


Fig. 1. A typical Filtered LFSR generator

of the output polynomial grows slowly over time. Since the attacker needs only the first output bit for each IV, it may be possible to apply the cube attack to such schemes, provided that they do not apply too many initialization rounds in which no output is produced. Results of the attack on simplified variants of Trivium that apply fewer initialization rounds are given in appendix B.

If the attacker is given more than one output bit in each execution of the stream cipher, he can slightly reduce the number of public variables required in the attack by summing the outputs of several polynomials p_i defining different output bits. This way he can get more than one linear equation for each maxterm during the preprocessing phase, and thus he can use fewer tweakable bits and use a smaller number of expensive restarts (which use many initialization steps) of the stream cipher during his attack.

An interesting observation is that unlike the case of other attacks, XOR'ing the outputs of several completely unrelated stream ciphers does not provide enhanced protection against cube attacks: If each one of the stream ciphers can be represented by a low degree multivariate polynomial, their XOR is also a low degree polynomial which can be attacked just as easily as the individual stream ciphers.

7 Conclusions

In this paper we introduced a new type of cryptanalytic attack and described some of its applications. It joins the rank of linear, differential, algebraic, and correlation attacks by being a generic attack that can be applied to many types of cryptographic schemes. We demonstrated its effectiveness by breaking (both in theory and with an actual implementation) a standard construction of a stream cipher which seems to be secure against all the previously known attacks. We

also used the attack to break simplified Trivium variants with complexity that is considerably lower than the complexity of previous known attacks. The attack is likely to be the starting point for a new area of research, and hopefully it will lead to a better understanding of what makes cryptosystems secure.

Acknowledgements. We would like to thank Shahram Khazaei, Willi Meier and Paul Crowley for independently verifying our results.

References

1. Ajwa, I.A., Liu, Z., Wang, P.S.: Gröbner bases algorithm. Technical report, ICM Technical Reports Series (ICM-199502-00) (1995)
2. Faugère, J.c.: A new efficient algorithm for computing gröbner bases (f4). Journal of Pure and Applied Algebra, 75–83 (1999)
3. Gwenole, A., Jean-Charles, F., Hideki, I., Mitsuru, K., Makoto, S.: Comparison Between XL and Groebner Basis Algorithms (2004)
4. Courtois, N.T., Klimov, A.B., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
5. Courtois, N., Patarin, J.: About the xl algorithm over gf(2). In: CT-RSA, pp. 141–157 (2003)
6. Yang, B.-Y., Chen, J.-M., Courtois, N.T.: On asymptotic security estimates in XL and gröbner bases-related algebraic cryptanalysis. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 401–413. Springer, Heidelberg (2004)
7. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
8. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
9. Arora, S.: Probabilistic checking of proofs: a new characterization of np. Journal of the ACM, 2–13 (1998)
10. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. Journal of Computer and System Sciences 47, 549–595 (1993)
11. Courtois, N.T., Meier, W.: Algebraic attacks on stream ciphers with linear feedback, pp. 345–359. Springer, Heidelberg (2003)
12. Golic, J.D.: On the security of nonlinear filter generators. In: Proceedings of the Third International Workshop on Fast Software Encryption, London, UK, pp. 173–188. Springer, Heidelberg (1996)
13. Courtois, N.T.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)
14. Englund, H., Johansson, T.: A new simple technique to attack filter generators and related ciphers. In: Selected Areas in Cryptography, pp. 39–53 (2004)
15. Golic, J.D., Clark, A., Dawson, E.: Generalized inversion attack on nonlinear filter generators. IEEE Trans. Comput. 49(10), 1100–1109 (2000)
16. Johansson, T., Jnsson, F.: Fast correlation attacks through reconstruction of linear polynomials, pp. 300–315. Springer, Heidelberg (2000)

17. Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: Fast Software Encryption, pp. 28–40. Springer, Heidelberg (1997)
18. Garey, M.R., Johnson, D.S.: Computers, and Interactibility. A guide to the theory of np-completeness. Bell Telephone Labratories, Incorporated
19. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks On Reduced-Round MD6 and Trivium. In: Fast Software Encryption. Springer, Heidelberg (2009)
20. estream: Ecrypt stream cipher project, http://www.ecrypt.eu.org/stream/
21. De Cannière, C., Preneel, B.: Trivium - a stream cipher construction inspired by block cipher design principles. estream, ecrypt stream cipher. Technical report, of Lecture Notes in Computer Science
22. Raddum, H.: Cryptanalytic results on trivium. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039, 2006 (2006), www.ecrypt.eu.org/stream/papersdir/2006/039.ps
23. Maximov, A., Biryukov, A.: Two trivial attacks on trivium. In: Selected Areas in Cryptography, pp. 36–55 (2007)
24. McDonald, C.C.C., Pieprzyk, J.: Attacking bivium with minisat, http://eprint.iacr.org/2007/040
25. Sönmez Turan, M., Kara, O.: Linear approximations for 2-round trivium. In: Proc. First International Conference on Security of Information and Networks (SIN 2007), pp. 96–105. Trafford Publishing (2007)
26. Englund, H., Johansson, T., Sönmez Turan, M.: A framework for chosen IV statistical analysis of stream ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)
27. Vielhaber, M.: Breaking one.fivium by aida an algebraic iv differential attack. Cryptology ePrint Archive, Report 2007/413
28. Fischer, S., Khazaei, S., Meier, W.: Chosen IV statistical analysis for key recovery attacks on stream ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)
29. Joux, A., Muller, F.: A chosen iv attack against turing. In: Selected Areas in Cryptography, pp. 194–207 (2003)
30. O'Neil, S.: Algebraic structure defectoscopy. Cryptology ePrint Archive, Report 2007/378
31. Juhani, M., Saarinen, O.: Chosen-iv statistical attacks on estream ciphers. In: Proceeding of SECRYPT 2006, pp. 260–266 (2006)

A Appendix: Extensions and Generalizations of Cube Attacks

Various generalizations of the cube attack can be successfully applied even to cryptosystems in which the attacker cannot find sufficiently many linear superpolys, and thus the original attack fails:

1. In block ciphers, the attacker can try to use a "meet in the middle" attack. Each bit in the middle of the encryption process can be described as either a polynomial in the plaintext and key bits, or as a polynomial in the ciphertext and key bits. Since the number of rounds is halved, the degree of each one of these polynomials may be the square root of the degree of the full polynomial

which describes the cipher (especially when the number of rounds is relatively small and these degrees did not hit their maximal possible values). Instead of equating the given ciphertext bits to their high degree polynomials, the attacker can equate the two low degree polynomials describing the two halves of the encryption and get an easier to solve master equation. This technique can also be extended to the case of double encryptions, where the attacker has the additional benefit that the secret key bits used in the two polynomials are disjoint. Note that the attacker can get multiple polynomial equations for each one of the bits in the middle or for any one of their polynomial combinations.

2. In some stream ciphers with many initialization rounds, it is difficult to find the low degree maxterms required for the attack. In these cases, given that the internal structure of the stream cipher in known, we can try a different approach: The attacker explicitly represents the state register bits as polynomials in terms of the public and private variables at some intermediate initialization round. Given this explicit representation, the attacker performs linearization on the private variables by replacing them with a new set of private variables, reducing the degrees of the state register bit polynomials. The values of the new set of private variables can then be recovered using the basic techniques of the cube attack. After the values of the new private variables are recovered, the attacker can solve for the original key by solving the equations obtained during linearization. If the cipher's state is invertible, or close to being invertible, the attacker can simply run the cipher backwards to recover the key, instead of solving equations. Note that a similar technique may also be used to attack block ciphers, given that the attacker can explicitly represent the polynomials at some intermediate encryption round.
3. The attacker can benefit from any system of linear equations (even if it has fewer than n equations), or from any system of nonlinear equations in which some of the variables occur linearly, by enumerating and testing only their smaller set of solutions.
4. The attacker can exploit ANY nonlinear superpoly he can find and compactly represent by guessing some of the secret variables in it and simplifying the result. In particular, guessing $n-1$ key bits will always suffice to turn any superpoly into an easy to solve linear equation in the remaining variable, and will thus result in an attack which is faster than exhaustive search, assuming that the evaluation of the superpoly is not too time consuming.
5. The attacker can try to solve the equations he can derive from the cube attack even when they are nonlinear, provided that their degrees are low enough. When m is large, the attacker can sum over many possible subsets of $d-1$ public variables, and get a highly overdefined system of nonlinear equations which might be solved by linearization or any other technique.
6. The attacker can easily recognize quadratic superpolys by a generalization of the BLR linearity test: The attacker randomly chooses vectors $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3} \in \{0,1\}^n$, and verifies that $p_{S(I)}[\mathbf{0}]+p_{S(I)}[\boldsymbol{x_1}]+p_{S(I)}[\boldsymbol{x_2}]+p_{S(I)}[\boldsymbol{x_3}]+p_{S(I)}[\boldsymbol{x_1}+\boldsymbol{x_2}]+p_{S(I)}[\boldsymbol{x_1}+\boldsymbol{x_3}]+p_{S(I)}[\boldsymbol{x_2}+\boldsymbol{x_3}]+p_{S(I)}[\boldsymbol{x_1}+\boldsymbol{x_2}+\boldsymbol{x_3}]=0$. Again, non quadratic functions are likely to be eliminated after a few tests. The test can

be further generalized to cubic functions and to polynomials of higher degree with the number of required function evaluations growing exponentially with the degree. The coefficient calculation for polynomials of higher degree can be generalized as well.

7. The attacker can use the cube attack even if he cannot compactly represent superpolys. In this case, the attacker decides on a subkey (i.e. a subset of private variables) whose value is guessed during the online phase. For each value of the subkey bits, the degree of the superpolys in the remaining private variables is likely to be reduced, and the attacker can compute and store them more efficiently. Since the cubes and corresponding superpolys are now key-dependant, they need to be computed and stored for each potential value of the subkey. This requires more preprocessing time and memory, but gives the attacker the extra flexibility of using different maxterms for each subset of keys.
8. The attacker is usually given more than one output bit, and thus more than one polynomial in the input bits. In addition to trying each one of them separately, he can test any polynomial combination of these polynomials and try to find some linear superpolys among these combinations.
9. Note that in the common mode of operation of stream ciphers in which $n = m$, and the secret key and public IV bits are XOR'ed together during the initialization step, the maximal possible degree of the polynomial representation of the scheme is n, whereas in the general case the maximal possible degree is $n + m$.
10. When the cryptographic scheme has an insufficient number of public variables (or none at all), we can recast the cube attack as a related key attack in which we are also allowed to flip some of the secret key bits during the online phase. By replacing some of the x_i variables by the combinations $x_i + v_i$, we may get linear $p_{S(I)}$ polynomials where none existed before.

B Appendix: Cube Attacks on Scaled-Down Trivium Variants

Trivium [21] is a stream cipher designed in 2005 by C. De Canni‘ere and B. Preneel and submitted to the Profile 2 (hardware) European project eSTREAM [20]. It has an exceptionally simple structure, which leads to very good performance in both hardware and software. Despite Trivium's simplicity, there are no substantial cryptanalytic results against it so far. Due to these outstanding qualities, Trivium was chosen as part of the portfolio for Profile 2 by the eSTREAM project.

B.1 Description of Trivium

Trivium's internal state consists of 288 bits stored in three NLFSRs of different lengths. In each round, each register is shifted by one bit. The feedback to each register consists of a non linear combination of bits from another register,

XORed with a bit from the same register. The output bit at the end of each round is a linear combination of six state bits, two taken from each register. During initialization, the 80-bit key is placed in the first register, and the 80-bit IV is placed in the second register. The other state bits are set to zero, except the last three bits in the third register, which are set to one. The state is then updated $4 \times 288 = 1152$ times without producing an output.

B.2 Previous Attacks

Trivium has a simple structure which led many cryptanalysts to try to attack it. Nevertheless, to this day, there are no attacks better than exhaustive search on the full version of Trivium. Due to Trivium's cryptanalytic resistance, scaled-down variants have been proposed and studied by cryptanalysts hoping to better understand the full-scale version. Two scaled-down variants named Bivium A and Bivium B were introduced in [22]. Both of these variants have an internal state composed of only 2 shift registers. Previous attacks on Trivium and its Bivium variants are summarized below:

- Raddum [22] developed an algorithm for solving sparse quadratic equations. The algorithm was used to break Bivium A in "about a day", and requires 2^{56} seconds to break Bivium B. The complexity of the attack applied to Trivium is 2^{164}.
- Maximov and Biryukov [23] developed a technique that can be applied to Bivium and Trivium. The technique involves guessing certain key bits and key bit products that reduce the Trivium quadratic equation system to a linear equation system that can be solved by linear algebra. The technique can be used to recover the state of Bivium B with complexity of $c \cdot 2^{36.1}$, and to recover the state of Trivium with complexity of $c \cdot 2^{83.5}$, where the constant c is the complexity of solving the system of linear equations.
- McDonald, Charnes, and Pieprzyk [24] showed that the MiniSat algorithm can be used to attack Bivium B with complexity of about 2^{56}.

Another family of scaled-down Trivium variants, assumes that fewer than 1152 initialization rounds are performed before producing an output. Previous attacks on Trivium variants with fewer than 1152 initialization rounds are summarized below:

- Turan and Kara [25] used linear cryptanalysis to give a linear approximation with bias 2^{-31} for Trivium with 288 initialization rounds.
- Englund, Johansson, and Turan [26] developed statistical tests and used them to show statistical weaknesses of Trivium with up to 736 initialization rounds. The basic idea is to use a statistical (rather than algebraic) variant of a cube attack, which selects an IV subset, examines all the keystream produced by assigning this subset all possible values, while keeping the other IV bits fixed. The key stream is viewed as a function of the selected IV subset variables, and statistical tests are performed to distinguish this function from a random one.

- Vielhaber [27] recovered 47 key bits of Trivium with 576 initialization rounds in negligible time. The key bits were recovered after some small IV special subsets were found, each one with the following property: The result of summing on some keystream bit produced by assigning a special subset all possible IV values, while keeping the other IV bits fixed, is equal to either one of the key bits or to the sum of two key bits. Note that this is a very special case of our cube attack, and it is not clear why the author imposed this unnecessary restriction.
- Fischer, Khazaei, and Meier [28] combined statistical tests with the method described in [27], and showed an attack on Trivium with 672 initialization rounds with complexity 2^{55}.

The last three attacks share their cube summing element with our attack, but then proceed in a different way, which does not apply efficient linearity testing to the resultant superpolys in order to find easy to solve linear equations. Our greatly improved cryptanalytic results for Trivium clearly demonstrate that cube attacks are more general, more efficient, and more powerful than these previous techniques.

B.3 The Attack

We summarize the results we obtained so far for various simplified variants of Trivium. All the maxterms and their associated linear equations were obtained by running the preprocessing phase of the cube attack in a high level language on a single PC over several weeks, and much better results can be expected by using a more optimized implementation on a cluster of more powerful computers.

- The best known attack on the variant which uses 672 initialization rounds is described by Fischer, Khazaei, and Meier in [28]. The authors attack this variant with complexity 2^{55}. We were able to find 63 linearly independent maxterms during the preprocessing phase of the cube attack on this variant (in fact, we found more, but the additional maxterms do not reduce the total complexity of the attack). All of the maxterms correspond to cubes of size 12. The maxterms and cubes are listed in Table 1 next to the summed output bit index. Both the key bit indexes and the IV bit indexes range from 0 to 79. The output bit index ranges from 672 to 685, hence the attacker needs up to 14 initial output bits produced by the cipher after the 672 key mixing rounds. Each of the maxterms passed at least 100 linearity tests, and thus the maxterm equations are likely to be correct for most keys. During the online phase of the cube attack, the attacker has to find the values of the linear equations defined by these maxterms by summing over the 63 cubes, of size 12. This requires a total of about 2^{18} chosen IVs. After the maxterm values are computed, the rest of the key can be recovered by exhaustive search with complexity 2^{17}. The total complexity of the attack is thus no more than 2^{19}, which is a big improvement compared to the best known attack. Note that the maxterms are very sparse, hence the complexity of the linear algebra in the preprocessing and online phases is negligible.

- We pushed the attack further by strengthening the Trivium variant to use 735 initialization rounds before producing an output. Currently, there is no known attack that is better than exhaustive search on this scaled-down Trivium variant. We were able to find 53 linearly independent maxterms corresponding to cubes of size 23 (again, we have more). The total complexity of the online phase of the attack is less than 2^{30}, which is much better than exhaustive search.
- Pushing the attack even further, we were able to find so far 35 maxterms for the stronger Trivium variant that uses 767 initialization rounds. The maxterms are listed in Table 2 in the appendix, next to the corresponding cubes. Most cubes are of size 29, but there are a few cubes of size ranging from 28 to 31. The complexity of the attack is 2^{45} since it is dominated by an exhaustive search for the $80-35=45$ missing key bits, after the values of the linear equations defined by these maxterms are computed. Computation on weaker variants shows that once a cube of a certain size that corresponds to a maxterm is found, we can expect to find many more cubes of the same size with linear superpolys. Thus, given more preprocessing resources, it is very likely that the online phase complexity of the attack can be reduced to about 2^{36}.

Our results show that even after many key mixing initializations rounds, Trivium is still breakable with complexity that is significantly faster than exhaustive search. We are still investigating the resistance of stronger Trivium variants to cube attacks and their generalizations.

B.4 Details of the New Cube Attacks on Scaled-Down Trivium Variants

Tables 1 and 2, list the maxterms, cube IV indexes, and output bit indexes for Trivium with 672 and with 767 initialization rounds respectively. In each one of the summations in Table 1 , all the public variables that do not belong to the cube were set to 0. In a few summations in Table 2, some public variables that do not belong to the cube were set to 1. These are specified in the last column. IV and key bits are indexed as in the original Trivium specification starting from 0 to 79 (e.g. key bits 65 and 68 and IV bits 68 and 77 determine the output bit with index 0).

Table 1. Maxterms for Trivium with 672 Initialization rounds

Maxterm Equation	Cube Indexes	Output Bit Index
1+x0+x9+x50	{2,13,20,24,37,42,43,46,53,55,57,67}	675
1+x0+x24	{2,12,17,25,37,39,46,48,54,56,65,78}	673
1+x1+x10+x51	{3,14,21,25,38,43,44,47,54,56,58,68}	674
1+x1+x25	{3,13,18,26,38,40,47,49,55,57,66,79}	672
1+x2+x34+x62	{0,5,7,18,21,32,38,43,59,67,73,78}	678
1+x3+x35+x63	{1,6,8,19,22,33,39,44,60,68,74,79}	677
x4	{11,18,20,33,45,47,53,60,61,63,69,78}	675
x5	{5,14,16,18,27,31,37,43,48,55,63,78}	677
x7	{1,3,6,7,12,18,22,38,47,58,67,74}	675
1+x8+x49+x68	{1,12,19,23,36,41,42,45,52,54,56,66}	676
x11	{0,4,9,11,22,24,27,29,44,46,51,76}	684
x12	{0,5,8,11,13,21,22,26,36,38,53,79}	673
x13	{0,5,8,11,13,22,26,36,37,38,53,79}	673
1+x14	{2,5,7,10,14,24,27,39,49,56,57,61}	672
x15	{0,2,9,11,13,37,44,47,49,68,74,78}	685
x16	{1,6,7,12,18,21,29,33,34,45,49,70}	675
x17	{8,11,15,17,26,23,32,42,51,62,64,79}	677
x18	{0,10,16,19,28,31,43,50,53,66,69,79}	676
x19	{4,9,10,15,21,24,32,36,37,48,52,73}	672
x20	{7,10,18,20,23,25,31,45,53,63,71,78}	675
1+x20+x50	{11,16,20,22,35,43,46,51,55,58,62,63}	675
1+x21+x66	{10,13,15,17,30,37,39,42,47,57,73,79}	673
x22	{2,4,21,23,25,41,44,54,58,66,73,78}	673
x23	{3,6,14,21,23,27,32,40,54,57,70,71}	672
1+x24	{3,5,14,16,18,20,33,56,57,65,73,75}	672
1+x28	{6,11,14,19,33,39,44,52,58,60,74,79}	676
x29	{1,7,12,18,21,25,29,45,46,61,68,70}	675
x30	{2,8,13,19,22,26,30,46,47,62,69,71}	674
x31	{3,9,14,20,23,27,31,47,48,63,70,72}	673
x32	{4,10,15,21,24,28,32,48,49,64,71,73}	672
x33	{2,4,6,12,23,29,32,37,46,49,52,76}	680
1+x34+x62	{0,5,7,13,18,21,32,38,43,59,73,78}	678
1+x35+x63	{1,6,8,14,19,22,33,39,44,60,74,79}	677
x36	{2,4,5,8,15,19,27,32,35,57,71,78}	677
x38+x56	{0,3,4,9,20,28,33,41,54,58,72,79}	678
1+x39+x57+x66	{8,11,13,17,23,25,35,45,47,54,70,79}	674
x40+x58+x64	{0,6,10,16,19,31,43,50,66,69,77,79}	676
1+x41	{2,15,17,20,21,37,39,44,46,56,67,73}	674
x42+x60	{1,16,20,22,34,37,38,53,58,69,71,78}	674
x43	{2,7,14,22,41,45,48,58,68,70,72,76}	673
x44+x62	{3,14,16,18,20,23,32,46,56,57,65,73}	672
1+x45+x64	{0,6,10,16,18,28,31,43,53,69,77,79}	676
x46+x55	{2,8,11,13,28,31,35,37,49,51,68,78}	684
x47	{5,8,20,32,36,39,45,51,65,69,76,78}	676
x48	{2,4,10,14,16,22,25,44,49,51,57,78}	678
x49+x62	{1,12,19,23,36,41,42,45,52,56,69,75}	676
x51+x62	{1,7,8,13,21,23,28,30,47,68,71,75}	674
x52	{5,8,9,12,16,18,23,40,44,63,66,70}	674
x53	{2,11,21,24,32,55,57,60,63,66,70,77}	675
1+x54+x60	{4,7,10,18,20,25,50,53,61,63,71,78}	675
x55+x64	{5,12,16,19,22,36,47,55,63,71,77,79}	674
1+x56	{4,9,18,21,23,27,32,38,43,58,67,69}	677
x57	{1,7,9,14,18,21,33,40,45,49,59,68}	675
1+x58	{2,6,12,13,19,23,30,48,55,59,69,79}	673
x60	{5,7,10,13,15,17,28,40,47,73,76,79}	681
x61	{13,21,24,39,42,46,48,51,55,61,72,78}	673
1+x62	{2,4,10,11,19,34,47,55,56,58,69,77}	674
x63	{5,7,10,15,17,35,40,47,52,57,76,79}	674
x64	{8,11,13,17,23,25,35,47,62,64,68,79}	673
x65	{2,3,13,15,19,29,32,37,39,51,76,79}	682
1+x66	{5,7,10,13,15,17,35,40,52,70,76,79}	678
1+x67	{5,20,24,29,33,35,37,39,63,65,74,78}	677
1+x68	{1,12,19,23,36,41,52,54,56,66,69,75}	676

Table 2. Maxterms for Trivium with 767 Initialization rounds

Maxterm Equation	Cube Indexes	Output	Variables set to 1
1+x0	{1,3,4,7,9,10,12,16,19,21,25,27,29,30,32,34,35,37,40,47,50,51,60,61,64,67,72,73,79}	769	
x3	{0,3,6,9,12,15,18,21,24,27,30,32,37,40,43,44,48,50,53,57,59,61,63,64,66,68,71,73,77,79}	773	{11}
x20	{1,3,5,7,10,14,18,20,22,23,26,30,36,38,42,43,44,45,47,49,52,54,60,63,69,71,72,73,78}	770	{53}
x22	{1,3,5,7,10,12,14,16,18,20,23,26,30,39,41,42,43,47,50,52,53,55,58,60,61,64,69,71,78}	769	
x23	{0,2,4,6,8,10,14,17,19,21,23,26,30,34,35,36,43,45,46,48,49,54,59,64,67,72,73,74,75,79}	767	
1+x29	{1,3,5,7,10,12,14,17,20,22,24,30,32,34,37,38,40,41,48,50,54,56,58,59,65,66,68,70,78}	774	
x30	{0,2,4,6,8,10,14,17,19,21,23,26,30,33,34,35,36,43,45,46,49,54,57,59,62,64,72,73,75,79}	773	{67}
1+x31	{0,2,4,6,8,10,13,14,17,19,21,23,26,30,31,34,35,36,37,42,53,60,61,64,66,69,72,73,77,79}	773	
x32	{0,2,4,6,8,10,14,17,19,21,23,25,26,27,30,32,34,43,44,53,58,63,68,70,71,72,75,78,79}	772	{33,37,38}
1+x33+x60+x66+x68	{1,3,5,7,10,14,18,20,23,26,30,35,37,39,40,41,44,48,49,51,54,58,59,60,61,64,70,75,77,78}	772	
1+x34	{1,3,5,7,10,12,14,16,17,20,24,28,30,33,34,36,40,42,45,46,51,52,54,56,62,66,70,77,78}	770	{76}
x35	{1,3,4,6,7,8,9,12,14,16,19,21,25,27,30,38,41,44,45,48,50,55,57,60,63,65,71,73,79}	769	
x36	{0,2,4,5,6,8,10,14,17,19,21,23,26,27,30,37,39,40,47,48,55,62,65,70,73,75,77,78,79}	768	{54}
x37	{1,3,5,7,10,12,14,16,17,20,24,26,30,32,35,37,41,45,46,54,58,60,64,67,68,69,70,72,78}	770	
x38	{0,2,4,6,8,10,14,17,19,23,25,26,30,34,36,38,40,42,44,53,56,57,60,63,69,72,73,75,79}	768	{39}
x41	{0,1,3,4,7,10,12,15,17,19,22,24,25,28,30,34,39,42,44,52,56,58,59,62,64,68,70,72,79}	773	{71}
1+x45	{1,3,5,7,10,12,14,16,18,20,22,23,26,30,33,39,42,43,47,50,52,53,55,58,60,64,71,77,78}	769	
1+x46	{1,3,5,8,11,14,16,17,19,21,23,26,27,29,30,32,36,38,42,44,45,49,51,53,59,60,63,64,75,76,78}	771	
x51	{0,2,4,6,8,10,14,17,19,23,26,30,33,38,39,41,43,46,47,50,54,58,59,60,62,63,64,71,72,77,79}	773	
1+x53+x57	{1,3,5,7,10,14,16,18,20,23,26,30,35,37,39,41,44,48,49,51,54,58,60,64,68,70,75,77,78}	773	{40,61}
x54	{0,2,4,6,8,10,14,17,19,23,26,30,33,38,39,41,43,46,50,54,59,60,61,62,63,64,70,74,77,79}	767	
1+x55	{1,3,5,7,10,12,14,17,18,20,24,27,30,33,36,38,40,41,44,53,56,59,61,66,68,72,75,76,78}	771	
x56	{1,3,5,7,9,12,14,16,19,21,23,25,27,30,35,37,40,51,56,62,63,64,67,69,71,74,75,76,79}	769	
1+x57	{1,3,5,7,10,12,14,17,20,24,30,32,34,37,38,40,48,50,52,54,56,57,58,59,63,66,68,70,78}	774	
x58	{0,2,4,6,8,10,14,17,19,21,23,26,30,33,36,43,45,48,49,54,57,59,62,64,67,72,74,75,79}	767	
x59+x65	{1,3,5,7,10,12,14,17,20,22,24,26,28,30,35,40,41,42,44,52,54,60,65,67,68,73,74,75,78}	773	
x60	{2,4,10,13,15,19,23,25,27,31,33,34,37,40,41,45,48,50,51,54,56,60,61,62,67,69,71,73,76}	770	
1+x60+x66	{1,3,4,5,7,9,12,16,19,21,25,27,30,32,33,35,38,40,43,45,47,51,55,57,59,60,62,75,79}	774	
x61	{3,5,11,14,16,20,24,26,28,32,34,35,38,41,42,46,49,51,52,55,57,61,62,63,68,70,72,74,77}	769	
x62	{1,3,5,7,10,12,14,17,20,22,24,26,28,30,35,40,41,42,44,47,52,54,65,66,67,68,73,75,78}	772	
1+x62+x68	{1,3,5,7,10,12,14,17,20,22,24,26,28,30,35,40,41,42,44,47,52,59,60,67,68,73,75,77,78}	773	
x63	{2,4,8,10,13,15,19,23,27,31,33,37,40,41,45,48,50,54,56,60,61,62,67,69,71,73,76,78}	770	
x64	{3,5,9,11,14,16,20,24,28,32,34,38,41,42,46,49,51,55,57,61,62,63,68,70,72,74,77,79}	769	
x65	{0,2,4,6,7,8,10,14,17,19,21,23,26,30,32,34,36,37,39,41,43,45,55,56,61,66,74,76,79}	767	
1+x67	{2,4,6,8,11,13,15,17,19,21,23,24,27,31,34,40,42,43,44,48,51,56,59,61,65,70,72,78,79}	768	

Smashing SQUASH-0

Khaled Ouafi* and Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. At the RFID Security Workshop 2007, Adi Shamir presented a new challenge-response protocol well suited for RFIDs, although based on the Rabin public-key cryptosystem. This protocol, which we call SQUASH-0, was using a linear mixing function which was subsequently withdrawn. Essentially, we mount an attack against SQUASH-0 with full window which could be used as a "known random coins attack" against Rabin-SAEP. We then extend it for SQUASH-0 with arbitrary window. We apply it with the proposed modulus $2^{1\,277} - 1$ to run a key recovery attack using 1 024 chosen challenges. Since the security arguments equally apply to the final version of SQUASH and to SQUASH-0, we challenge the blame-game argument for the security of SQUASH. Nevertheless, our attacks are inefficient when using non-linear mixing so the security of SQUASH remains open.

Keywords: RFID, cryptanalysis, MAC.

1 The SQUASH Algorithm

RFID tags use challenge-response protocols in which a reader sends a random challenge to a RFID tag to which this latter responds by computing the output of an algorithm (generally a MAC) fed with the challenge and a unique secret key. The reader then goes through a database containing a list of secrets associated with the identity of each tag to find the matching secret and thus the tag's identity. Due to computation and power constraints, most of the primitives proposed for RFID tags rely on symmetric key primitives since they offer competitive throughput, compared to public-key primitives which require much more transistors to be implemented as well as longer computation time.

At the RFID Security Workshop 2007, Adi Shamir [3] presented the SQUASH algorithm, a message authentication code (MAC) which, although based on the Rabin public-key cryptosystem, performs very well on benchmarks. In addition to this, it offers some kind of provable security based on the hardness of factoring large integers. This proof is used as a "safety net" as no attack using the modulus factors is known so far.

Essentially, SQUASH consists of a public hard-to-factor modulus N of ℓ bits and a r-bit length key which is also the length of the challenge. The algorithm is very simple as it only:

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* Supported by a grant of the Swiss National Science Foundation, 200021-119847/1.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 300–312, 2009.

– mixes the challenge and the secret key using a mixing function;
– converts it in a number;
– squares it modulo N;
– truncates the result to a specific window of bits.

There was essentially two versions of SQUASH. The first one considered a randomized version of the Rabin cryptosystem in which the square was hidden by adding a random multiple of N to avoid modulo N reduction, but which required a full-size window (i.e. no truncation at the end). The second one (preferred for efficiency reasons) considered a small window in the middle of the ℓ-bit result. Contrarily to the Rabin cryptosystem, SQUASH does not need to be invertible. So, the factorization of the modulus N does not need to be known by any participant. This motivated the recommendation of [4] to use Mersenne numbers (numbers of the form $N = 2^\ell - 1$) or the more general Cunningham project numbers (numbers of the form $N = a \cdot b^c \pm d$ for small a, b, c, d) whose factorization is unknown so far. Any other technical details regarding SQUASH are not relevant for our analysis.

Through this paper, we denote by R, K, C, and T the response, key, challenge, and truncation function of SQUASH respectively. The function SQUASH will simply consist of the following:

$$R = T\left(\left(\sum_{i=0}^{\ell-1} 2^i \times f_i(K, C)\right)^2 \bmod N\right),$$

where the f_i's are Boolean functions and the truncation function T is defined by

$$T(x) = \left\lfloor \frac{x \bmod 2^b}{2^a} \right\rfloor.$$

By expanding the square, we obtain:

$$R = T\left(\left(\sum_{i=0}^{\ell-1}\sum_{i'=0}^{\ell-1} 2^{i+i'} \times f_i(K, C) f_{i'}(K, C)\right) \bmod N\right) \tag{1}$$

The version of SQUASH presented in 2007, which we call SQUASH-0, uses a mixing function f expanding (using a linear feedback shift register) the XOR of the key and the challenge. It was subsequently updated in the version available on [3] following a private comment by Vaudenay. This comment was about a total break on the first version, i.e. the variant using no truncation. Nevertheless, the version published in [4] suggests to use a concrete non-linear mixing function. In this paper, we first present the attack by Vaudenay and apply it to any mixing function of form $g(K) \oplus L(C)$. Then, for L linear, we use discrete Fourrier transform to propose a variant of the attack and we generalize it to the second variant of SQUASH-0 using an arbitrary window.

Consequently, we restrict to the special case where there exists Boolean functions g_i and L_i such that

$$f_i(K, C) = g_i(K) \oplus L_i(C)$$

for all i, K, and C, where $\oplus$ denotes the exclusive or (XOR) operation. This is the case for the Rabin-SAEP encryption [1] where K plays the role of the plaintext and C plays the role of the random coins. In Sections 3 and 4, we further assume that the L_i's are linear in the sense that $L_i(a \oplus b) = L_i(a) \oplus L_i(b)$ for any a and b.

In what follows we first consider in Section 2 Vaudenay's passive attack against this algorithm with a full-size window. Namely, we assume the adversary gets the full Rabin encryption but no linearity in the L_i's. This translates into a "known random coins attack" against Rabin-SAEP. It works with complexity $O(\ell^2 r^6)$ and $O(r^2)$ known challenges. Next, we study an active variant of this attack in the linear case in Section 3 working in complexity $O(\ell r^2 \log r / \log\log r)$ and $O(r^2/\log\log r)$ chosen challenges. Finally, we apply the variant to the case of an arbitrary window in Section 4. Our final attack has a complexity of $O(\ell r^2 \log r)$ and uses $O(r^2)$ chosen challenges. It works when the window is large enough, namely $b - a > 4\log_2 r - 2$. However, in the case of Mersenne numbers, those figures boil down to a complexity of $O(r^2 \log r)$ and $O(r^2/\ell)$ chosen challenges, and the condition is relaxed to $b - a > 2\log_2 r - \log_2 \ell - 1$. When the window is smaller, there is still room for further improvements. We conclude that SQUASH-0 is dead broken and that the security proof in SQUASH is incorrect if factoring is hard. However, the security of SQUASH is still open.

2 Passive Attack with Full-Size Window

The goal of a passive total-break attack, is for an adversary to derive the secret key from challenge-response samples only. In what follows, we denote $k_i = (-1)^{g_i(K)}$ and $c_i = (-1)^{L_i(C)}$. We have

$$f_i(K, C) = \frac{1 - k_i c_i}{2}$$

which is linear (in the sense of $\mathbb{Z}$) in terms of k_i with coefficients known to the adversary. By expanding (1) we derive

$$R = \frac{1}{4}\sum_{i,i'} 2^{i+i'} c_i c_{i'} k_i k_{i'} - \frac{2^\ell - 1}{2}\sum_i 2^i c_i k_i + \frac{(2^\ell - 1)^2}{4} \bmod N \qquad (2)$$

when no truncation T is used.

A first attack consists of collecting enough equations of this form and solving them, e.g. by linearization or re-linearization [2]. Simple lineralization consists of expressing $k_i k_{i'}$ as a new unknown and solving linear equations. We get $\frac{r(r+1)}{2}$ unknowns and a solving algorithm of complexity $O(\ell^2 r^6)$ (as for $O(r^6)$ multiplications with complexity $O(\ell^2)$) after collection of $O(\ell r^2)$ bits (as for $O(r^2)$ samples of $O(\ell)$ bits). Since $k_i k_{i'} = \pm 1$ which is unexpectedly small, we can also consider algorithms based on lattice reduction using $O(\ell)$ samples only. The attack works even if the L_i's are not linear. In the Rabin-SAEP case, we obtain a "known random coins attack" in which an adversary can request many encryptions of the same plaintext and get the random coins with. His purpose

is to recover the plaintext. However, for ℓ resp. r in the order of magnitude of 2^{10} resp. 2^6, complexities are still very high.

Interestingly, we note that when N is a Mersenne number then $N = 2^\ell - 1$ so Equation (2) simplifies by getting rid of r unknowns. Therefore, we have $\frac{r(r-1)}{2}$ unknowns instead of $\frac{r(r+1)}{2}$.

3 Active Attack with Full-Size Window

Let $C_1, \ldots, C_d$ be a set of d random challenges, given an integer d to be later discussed. Let U_i be the d-bit vector with coordinate $L_i(C_j)$, $j = 1, \ldots, d$. We consider an active attack making 2^d chosen challenges. Given a d-bit vector x, we define the challenge $C(x) = \bigoplus_j x_j C_j$ and $R(x)$, the response obtained after submitting this challenge. Given a real function $\varphi(x)$ and a d-bit vector V, we further recall the multidimentional discrete Fourrier transform of a function φ with group $\mathbb{Z}_2^d$:

$$\hat{\varphi}(V) = \sum_x (-1)^{x \cdot V} \varphi(x)$$

where $x \cdot V$ denotes the scalar dot product of vectors x and V. Thanks to the linearity of L_i, we have

$$c_i(x) = (-1)^{L_i(C(x))} = (-1)^{\bigoplus_j x_j L_i(C_j)} = (-1)^{x \cdot U_i}$$

Using the following:

- $\varphi(x) = 1 \Longrightarrow \hat{\varphi}(V) = 2^d \times 1_{V=0}$;
- $\varphi(x) = (-1)^{x_i U_i} \Longrightarrow \hat{\varphi}(V) = 2^d \times 1_{U_i = V}$;
- $\varphi(x) = (-1)^{x_i (U_i \oplus U_j)} \Longrightarrow \hat{\varphi}(V) = 2^d \times 1_{U_i \oplus U_j = V}$;

we deduce from Equation (2) that

$$\begin{aligned} \hat{R}(V) = \frac{1}{4} \sum_{i,i'} 2^{i+i'} \left(\sum_x (-1)^{x \cdot (U_i \oplus U_{i'} \oplus V)} \right) k_i k_{i'} \\ - \frac{2^\ell - 1}{2} \sum_i 2^i \left(\sum_x (-1)^{x \cdot (U_i \oplus V)} \right) k_i \\ + \frac{(2^\ell - 1)^2}{4} \sum_x (-1)^{x \cdot V} \pmod N. \end{aligned} \tag{3}$$

3.1 First Method

Let I be an integer between 0 and $\ell - 1$. We assume that $C_1, \ldots, C_d$ are chosen such that

- for all j we have $L_I(C_j) = 0$;
- for every $i < i'$ there exists j such that $L_i(C_j) \neq L_{i'}(C_j)$.

In terms of U_i's, the hypotheses translate into observing that all the U_i's are pairwise different and that one of these vectors is the vector containing only 0's. Clearly, we can find these vectors by using an incremental algorithm to select C_j's in the hyperplane defined by $L_I(C) = 0$. If we generate d random vectors in the hyperplane, under heuristic assumptions, the probability that the condition is fulfilled is roughly $e^{-\ell^2 2^{-d-1}}$ which is constant for $d = 2\lceil \log_2 \ell \rceil$ and equal to $e^{-1/2}$.

By (3), thanks to the hypotheses we obtain

$$\hat{R}(0) = 2^{d-2} \sum_i 2^{2i} k_i^2 - 2^{d+I-1}(2^\ell - 1)k_I + \frac{2^d(2^\ell - 1)^2}{4} \pmod N$$

but since $k_i^2 = 1$ for all i we obtain

$$\hat{R}(0) = 2^{d-1}(2^\ell - 1)\left(\frac{2}{3}2^\ell - \frac{1}{3} - 2^I k_I\right) \pmod N$$

We can thus deduce k_I when N is not a Mersenne number. This means that recovering the key requires $O(r\ell^2)$ chosen challenges and complexity $O(r\ell^3)$. Clearly, we can trade data complexity against time complexity.

The Mersenne case. If N is a Mersenne number $N = 2^\ell - 1$, as suggested in [4], the above expression vanishes so we have to make a specific treatment. Actually, we have

$$R = \frac{1}{4} \sum_{i,i'} 2^{i+i'} c_i c_{i'} k_i k_{i'} \bmod N$$

By changing the assumption about the C_i's to

- there is a unique $I < J$ pair such that $U_I = U_J$

we obtain

$$\hat{R}(0) = 2^{I+J+d-1} k_I k_J \pmod N$$

so we can still adapt the attack. Other kinds of Cunningham numbers $N = a^b \pm c$, as suggested in [4], work like in the previous case.

3.2 Second Method

We now relax the assumption about the C_i's. By taking a value V such that

- $V \notin \{0, U_0, U_1, \ldots, U_{\ell-1}\}$
- there exists a unique $\{I, J\}$ pair such that $V = U_I \oplus U_J$

then (3) simplifies to

$$\hat{R}(V) = 2^{I+J-1+d} k_I k_J \pmod N$$

so we can deduce the value of $k_I k_J$.

The advantage of this method is that from the same set of challenges we can derive many equations of the form $k_I k_J = b$ (which are indeed linear equations) for all I and J such that $V = U_I \oplus U_J$ satisfies the above conditions. With random C_i's, the expected number of such equations is roughly $\frac{1}{2}\ell^2 e^{-\ell^2 2^{-d-1}}$ so for $d \approx 2\log_2 \ell$ we obtain enough equations to recover all bits of K using $O(\ell^2)$ chosen challenges and complexity $O(\ell^3 \log \ell)$.

3.3 Generalization

We can further generalize this attack by taking all values V which are either 0 or equal to some U_I or to some $U_I \oplus U_J$ but without requiring unicity of I or $\{I, J\}$. In general, we obtain an equation which may involve several k_I or $k_I k_J$ as Equation (3) simplifies to

$$\hat{R}(V) = \sum_{\{I,J\}:U_I \oplus U_J = V} 2^{I+J-1+d} k_I k_J - \sum_{I:U_I=V} (2^\ell - 1)2^{I+d-1} k_I + (2^\ell - 1)^2 2^{d-2} 1_{V=0} \pmod N.$$

Provided that the number of monomials is not too large, the only correct ± 1 assignment of the monomials leading to an expression matching the $\hat{R}(V)$ value can be isolated.

Using $d = \log_2 \frac{r(r+1)}{2}$ we obtain only one unknown per equation on average so we can recover all key bits with complexity $O(\ell r^2 \log r)$ using $O(r^2)$ chosen challenges. We can still slightly improve those asymptotic figures.

Let ℓm be the complexity of getting the matching ± 1 assignments in one equation (i.e. m is the complexity in terms of modulo N additions). The complexity of the Fourier transform is $O(\ell d 2^d)$, so the complexity of the algorithm becomes $O(\ell(d+m)2^d)$. The average number of unknowns per equation is $r^2 2^{-d-1}$. By using an exhaustive search strategy to solve the equation we obtain $\log_2 m \approx r^2 2^{-d-1}$. With $d = 2\log_2 r - \log_2 \log_2 \log r - 1$ we have $m = \log r$ and we finally obtain a complexity of $O(\ell r^2 \log r / \log\log r)$ with $O(r^2 / \log\log r)$ chosen challenges.

We could view the equation as a knapsack problem and use solving algorithms better than exhaustive search. For instance, we can split the equation in two halves and use a claw search algorithm. The effect of this strategy leads us to $\log_2 m \approx \frac{1}{2} r^2 2^{-d-1}$ and we reach the same asymptotic complexity. However the practical benefit may be visible as the following example shows.

Example 1. SQUASH with no truncation is trivially broken if we can factor the modulus N so it should be at least of 1 024 bits. As an example, for $\ell = 1\,024$ and r arbitrary (up to ℓ) we can take $d = 14$ so roughly $2^d \approx 16\,000$ chosen challenges. We obtain at most $\frac{\ell(\ell+1)}{2} 2^{-d} \approx 32$ unknowns per equation on average. Using claw search algorithm will work with 2^{16} numbers in memory and 2^{16} iterations to recover 32 bits of the key for each equation.

The Mersenne case. Finally, another nice thing with Mersenne numbers is that the equation further simplifies to

$$\hat{R}(V) = \sum_{n=0}^{\ell-1} 2^n \sum_{\substack{\{I,J\}: \\ U_I \oplus U_J = V, I+J-1+d \bmod \ell = n}} k_I k_J \pmod N. \tag{4}$$

So, if the set of $\{I, J\}$'s sparsely spread on $(U_I \oplus U_J, (I + J - 1 + d) \bmod \ell)$ pairs, the knapsack is nearly super-increasing and we can directly *read* all $k_I k_J$ bits in the table of all $\hat{R}(V)$'s. That is, m is constant. With $d = 2\log_2 r - \log_2 \ell - 1$

we roughly have ℓ unknowns per equation and we can expect this phenomenon. So, we obtain a complexity of $O(r^2 \log r)$ with $O(r^2/\ell)$ chosen challenges. For instance, with $N = 2^{1\,277} - 1$ and $r = 128$ we can take $d = 3$ so that 8 chosen challenges are enough to recover all bits. With $r = \ell$ we can take $d = 10$ so that 1 024 chosen challenges are enough.

Example 2. As a toy example we consider a SQUASH instance with $N = 13 \times 19 = 247$, $\ell = 8$, and $r = 4$ along with the function

$$f(K, C) = (K \oplus C) \| (K \oplus C)$$

with $d = 2$.

If $K = \mathsf{0x9}$, we have $k_0 = k_3 = k_4 = k_7 = -1$ and $k_1 = k_2 = k_5 = k_6 = +1$. We take 2 random values for the C_i's: $C_1 = \mathsf{0x2}$ and $C_2 = \mathsf{0xa}$. Here is a table for the $C(x)$ values:

x	$C(x)$	$f(K, C(x))$	$R(x)$
00	0x0	0x99 = 153	191
10	0x2	0xbb = 187	142
01	0xa	0x33 = 51	131
11	0x8	0x11 = 17	42

V	$\hat{R}(V)$
00	506
10	138
01	160
11	−40

The U_i's are

$$U_0 = U_4 = 00, \quad U_1 = U_5 = 11, \quad U_2 = U_6 = 00, \quad U_3 = U_7 = 01.$$

We sort monomials following the corresponding values of V as follows:

- $V = 00$: k_0, k_4, k_2, k_6, k_0k_4, k_1k_5, k_2k_6, k_3k_7, k_0k_2, k_0k_6, k_2k_4, k_4k_6,
- $V = 01$: k_3, k_7, k_0k_3, k_0k_7, k_3k_4, k_4k_7, k_2k_3, k_2k_7, k_3k_6, k_6k_7,
- $V = 10$: k_1k_3, k_1k_7, k_3k_5, k_5k_7,
- $V = 11$: k_1, k_5, k_0k_1, k_0k_5, k_1k_4, k_4k_5, k_1k_2, k_1k_6, k_2k_5, k_5k_6.

Due to the structure of f we know that $k_0 = k_4$, $k_1 = k_5$, $k_2 = k_6$ and $k_3 = k_7$ so the list simplifies (modulo $N = 247$) to:

$$\begin{aligned}\hat{R}(00) = {} & \left(2^{0+4+1} + 2^{1+5+1} + 2^{2+6+1} + 2^{3+7+1}\right) \\ & + \left(2^{0+2+1} + 2^{0+6+1} + 2^{2+4+1} + 2^{4+6+1}\right) k_0k_2 \\ & -(2^8 - 1)\left(2^{0+1} + 2^{4+1}\right) k_0 - (2^8 - 1)\left(2^{2+1} + 2^{6+1}\right) k_2 \\ & +(2^8 - 1)^2 + \frac{1}{3}(2^{16} - 1)\end{aligned}$$

$$\begin{aligned}
\hat{R}(10) &= \left(2^{1+3+1} + 2^{1+7+1} + 2^{3+5+1} + 2^{5+7+1}\right) k_1 k_3 \\
\hat{R}(01) &= -(2^8 - 1)\left(2^{3+1} + 2^{7+1}\right) k_3 \\
&\quad + \left(2^{0+3+1} + 2^{0+7+1} + 2^{3+4+1} + 2^{4+7+1}\right) k_0 k_3 \\
&\quad + \left(2^{2+3+1} + 2^{2+7+1} + 2^{3+6+1} + 2^{6+7+1}\right) k_2 k_3 \\
\hat{R}(11) &= \left(2^{0+1+1} + 2^{0+5+1} + 2^{1+4+1} + 2^{4+5+1}\right) k_0 k_1 \\
&\quad + \left(2^{1+2+1} + 2^{1+6+1} + 2^{2+5+1} + 2^{5+6+1}\right) k_1 k_2 \\
&\quad - (2^8 - 1)\left(2^{1+1} + 2^{5+1}\right) k_1
\end{aligned}$$

which yields

$$\begin{aligned}
\hat{R}(00) &= 176 + 89k_0k_2 - 25k_0 - 100k_2 \pmod{247} \\
\hat{R}(10) &= 109k_1k_3 \pmod{247} \\
\hat{R}(01) &= -200k_3 + 178k_0k_3 + 218k_2k_3 \pmod{247} \\
\hat{R}(11) &= 168k_0k_1 + 178k_1k_2 - 50k_1 \pmod{247}
\end{aligned}$$

The only values of k_0 and k_2 leading to $\hat{R}(00) = 506 \pmod N$ is $k_0 = -1$ and $k_2 = +1$. Similarly, k_1 and k_3 lead to $\hat{R}(10) = 138 \pmod N$ if and only if $k_1 = -k_3$. Using $\hat{R}(01) = 160 \pmod N$, we deduce $k_3 = -1$. From these values, we recover the key $K = 9$.

4 Application to Limited Windows

In what follows we let S denote the Rabin encryption. We assume that S is truncated to a window defined by

$$T(x) = \left\lfloor \frac{x \bmod 2^b}{2^a} \right\rfloor$$

so that $R = T(S)$. Our analysis from Section 3 still applies when R is replaced by S. However, S is not directly available to the adversary.

Since it is not clear how to break this variant of SQUASH even when N can be factored, ℓ could be much smaller than usual values for modulo bit-length. Indeed, [4] suggested an aggressive $\ell = 128$ with the Mersenne number $N = 2^{128} - 1$, $a = 48$, $b = 80$, and $r = 64$.

4.1 First Method

First, by observing that for any $e_1, \ldots, e_n \in \mathbb{Z}_N$ we have

$$\left(\sum_{i=1}^{n} e_i \bmod N\right) \bmod 2^b = \left(\left(\sum_{i=1}^{n} e_i \bmod 2^b\right) + \left(-\beta N \bmod 2^b\right)\right) \bmod 2^b \quad (5)$$

for some $0 \leq \beta < n$ thus

$$T\left(\sum_{i=1}^{n} e_i \bmod N\right) = \left(\sum_{i=1}^{n} T(e_i) + T(-\beta N) + \alpha\right) \bmod 2^{b-a}$$

for some $0 \leq \alpha \leq n$.

We now apply the previous attack (first method) with $n = 2^d$ and the list of all d-bit vectors x. We use $e_i = S(x)$ corresponding to the challenge $C(x)$. We deduce

$$T\left(\hat{S}(0) \bmod N\right) = \left(\hat{R}(0) + T(-\beta N) + \alpha\right) \bmod 2^{b-a}.$$

Let

$$y_I^b = 2^{d-1}(2^\ell - 1)\left(\frac{2}{3}2^\ell - \frac{1}{3} - 2^I b\right) \bmod N$$

for $b = \pm 1$. Based on the previous attack we obtain

$$T(y_I^{k_I}) = \left(\hat{R}(0) + T(-\beta N) + \alpha\right) \bmod 2^{b-a}$$

for some $\alpha \in [0, 2^d]$ and $\beta \in [0, 2^d - 1]$. The probability that there exists α and β such that $T(y_I^{-k_I})$ matches the right-hand side of the equation is at most 2^{2d-r}, so for $2d + 1 < r$ it is likely that we can deduce k_I.

The Mersenne case. When N is a Mersenne prime number, the $T(-\beta N)$ expression simplifies to $T(\beta)$. This is always 0 for $d \leq a$ and it can be integrated in the α in $T(-\beta N) + \alpha$ in other cases: all $T(-\beta N) + \alpha$ values are numbers in the $\left[0, 2^d + \left\lfloor \frac{2^d - 1}{2^a} \right\rfloor\right]$ range. In what follows we assume that $d \leq a$ for simplicity.

We now use $y_{I,J}^b = 2^{I+J+d-1} b \bmod N$ and, with the updated hypotheses on the C_j vectors, we obtain

$$T(y_{I,J}^{k_I k_J}) = \left(\hat{R}(0) + \alpha\right) \bmod 2^{b-a}$$

for some α in the $[0, 2^d]$ range. Note that $T(y_{I,J}^{-1}) = \overline{T(y_{I,J}^{+1})}$ and that

$$\begin{aligned} T(y_{I,J}^{+1}) &= T\left(2^{(I+J+d-1) \bmod \ell}\right) \\ &= \begin{cases} 2^{((I+J+d-1) \bmod \ell) - a} & \text{if } a \leq (I+J+d-1) \bmod \ell < b \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

This is enough to deduce $k_I k_J$ for (I, J) pairs such that there is no α for which $T(y_{I,J}^{-k_I k_J})$ matches the right-hand side. Thus we can recover $k_I k_J$.

4.2 Second Method

Similarly to (5), if $\varepsilon_i = \pm 1$ and $\varepsilon_1 + \cdots + \varepsilon_n = 0$ we have

$$\left(\sum_{i=1}^{n} \varepsilon_i e_i \bmod N\right) \bmod 2^b = \left(\left(\sum_{i=1}^{n} \varepsilon_i (e_i \bmod 2^b)\right) - \beta N \bmod 2^b\right) \bmod 2^b$$

for some $|\beta| < \frac{n}{2}$ thus

$$T\left(\sum_{i=1}^{n} \varepsilon_i e_i \mod N\right) = \left(\sum_{i=1}^{n} \varepsilon_i T(e_i) + T(-\beta N) + \alpha\right) \bmod 2^{b-a}$$

for some $-\frac{n}{2} < \alpha \leq \frac{n}{2}$. We deduce that for each V there exist α and β verifying

$$T\left(\hat{S}(V) \bmod N\right) = \left(\hat{R}(V) + T(-\beta N) + \alpha\right) \bmod 2^{b-a}.$$

With the appropriate conditions on the set of challenges we obtain

$$T\left(\left(2^{I+J-1+d} k_I k_J\right) \bmod N\right) = \left(\hat{R}(V) + T(-\beta N) + \alpha\right) \bmod 2^{b-a}$$

which can be used to recover $k_I k_J$.

In the Mersenne case with $d < a$, $T(-\beta N)$ is either 0 or $2^{b-a} - 1$ depending on the sign of β so we have a single additional value for $T(-\beta N) + \alpha$ which is $-\frac{n}{2}$. Since we will always take $d < a$ we omit the other cases.

4.3 Generalization

More generally, for all V there exists α and β such that

$$\begin{aligned} T\Bigg(\Bigg(&\sum_{\{I,J\}:U_I \oplus U_J = V} 2^{I+J-1+d} k_I k_J \\ &- \sum_{I:U_I=V} (2^\ell - 1) 2^{I+d-1} k_I + (2^\ell - 1)^2 2^{d-2} 1_{V=0}\Bigg) \bmod N\Bigg) \\ = &\left(\hat{R}(V) + T(-\beta N) + \alpha\right) \bmod 2^{b-a} \end{aligned}$$

with either $-\frac{n}{2} < \alpha \leq \frac{n}{2}$ and $|\beta| < \frac{n}{2}$ for $V \neq 0$ or $0 \leq \alpha \leq n$ and $0 \leq \beta < n$ for $V = 0$.

Our attack strategy can now be summarized as follows.

1. Take a value for d. Make a table of all $T(-\beta N) + \alpha$ values. This table has less than 2^{2d} terms ($2^d + 1$ in the Mersenne case) and can be compressed by dropping the d least significant bits (corresponding to the α part). In the Mersenne case, it can be compressed to nothing as numbers of form $T(-\beta N) + \alpha$ are all in the $[-2^{d-1}, 2^{d-1}]$ range modulo 2^{b-a}.
2. Pick d challenges at random and query all the 2^d combinations $C(x)$. Get the responses $R(x)$.
3. Compute the discrete Fourier transform $\hat{R}$ in $O(\ell d 2^d)$.
4. For each V, try all ± 1 assignments of occurring unknowns in $\hat{S}(V)$ and keep all those such that $T(\hat{S}(V) \bmod N) - \hat{R}(V)$ is of form $T(-\beta N) + \alpha$.

Again, this attack uses $O(2^d)$ chosen challenges and a complexity of $O(\ell(d + 2^{s2^{-d}})2^d)$ where s is the number of unknowns, i.e. $s = \frac{r(r+1)}{2}$ resp. $s = \frac{r(r-1)}{2}$ in the Mersenne case. The remaining question is whether all wrong assignments are discarded.

For a given equation, each of the $2^{s2^{-d}}$ wrong assignments is discarded with probability $2^{2d-(b-a)}$ resp. $2^{d-(b-a)}$. Thus, if $b - a > 2d + s2^{-d}$ resp. $b - a > d + s2^{-d}$ they can all be filtered out. The minimum of the right-hand side is $2\log_2 s + 2\log_2 \frac{e\ln 2}{2}$ resp. $\log_2 s + \log_2(e\ln 2)$ and reached by $d = \log_2 s + \log_2 \frac{\ln 2}{2}$ resp. $d = \log_2 s + \log_2 \ln 2$. By taking this respective value for d we have $O(r^2)$ chosen challenges and a complexity of $O(\ell r^2 \log r)$, and the condition becomes $b - a > 4\log_2 r + 2\log_2 \frac{e\ln 2}{2} - 2$ resp. $b - a > 2\log_2 r + \log_2(2e\ln 2)$. If $b - a > 4\log_2 r - 2$ resp. $b - a > 2\log_2 r$ this condition is always satisfied.

Example 3. We continue our toy example with $a = 1$ and $b = 7$ (i.e. we drop the least and most significant bits after the Rabin encryption). We still use 4 chosen challenges which are the linear combinations of $C_1 = 2$ and $C_2 = 10$ and compute the $\hat{R}(V)$ values (without any modular reduction). Here is a table for the $R(x)$ and $\hat{R}(V)$ values:

x	$C(x)$	$f(K, C(x))$	$R(x)$
00	0x0	0x99 = 153	$T(191) = 31$
10	0x2	0xbb = 187	$T(142) = 7$
01	0xa	0x33 = 51	$T(131) = 1$
11	0x8	0x11 = 17	$T(42) = 21$

V	$\hat{R}(V)$
00	60
10	4
01	16
11	44

The possible values of $T(-\beta N)$ for $|\beta| < \frac{n}{2}$ are in $\{0, 4, -5\}$. The possible values for $T(-\beta N) + \alpha$ are in $[-7, 5] - \{-3\}$. We take the equation $\hat{R}(10) = 4$. The possible values for $\hat{R}(10) + T(-\beta N) + \alpha$ modulo 64 are in $[-3, 9] - \{1\}$. On the other hand, $T(109k_1k_3 \bmod N)$ can only be 5 (for $k_1k_3 = -1$) or -10 (for $k_1k_3 = +1$) so we deduce $k_1k_3 = -1$. Similarly, for $V = 01$ we obtain

$$T(178k_0k_3 + 218k_2k_3 - 200k_3 \bmod N) \in [9, 21] - \{13\}$$

The 8 possible values for $T(178k_0k_3 + 218k_2k_3 - 200k_3 \bmod N)$ are

$k_0k_3\ k_2k_3\ k_3$	$+++$	$++-$	$+-+$	$+--$	$-++$	$-+-$	$--+$	$---$
T	34	51	3	16	43	56	8	25

We deduce $k_3 = -1$, $k_0k_3 = +1$, $k_2k_3 = -1$ as the only possible assignment. Again, we recover $K = 9$.

The Mersenne case. Finally, the Mersenne case can simplify further using Equation (4). We take $d = 2\log_2 r - \log_2 \ell - 1$ and run the attack with $O(r^2/\ell)$

chosen challenges and complexity $O(r^2 \log r)$. Assuming that all unknowns $k_I k_J$ sparsely spread on $(U_I \oplus U_J, (I + J - 1 + d) \bmod \ell)$ pairs then $T(\hat{R}(V) \bmod N)$ yields $b-a-d$ useful bits with roughly one $k_I k_J$ per bit and ends with d garbage bits coming from $T(-\beta N) + \alpha$. So, we can directly *read* the bits through the window and it works assuming that $b - a > d$, which reads $b - a > 2 \log_2 r - \log_2 \ell - 1$.

Example 4. We now use the parameters from [4]: $\ell = 128$, $N = 2^{128} - 1$, $a = 48$, $b = 80$. Although [4] suggested a 64-bit secret with non-linear mixing we assume here that the mixing is of form $f = g \oplus L$ with linear L but that g expands to $r = 128$ secret bits (possibly non-linearly). We have $s = 8\,128$ unknowns of form $k_i k_{i'}$. With $d = 10$ we obtain $1\,024$ vectors V so we can expect to find 8 unknowns in each equation. Equations are of form

$$T(\hat{S}(V) \bmod N) = \left(\hat{R}(V) + T(-\beta N) + \alpha\right) \bmod 2^{b-a}$$

where $(T(-\beta N) + \alpha) \bmod 2^{b-a}$ is in the range $[-2^9, 2^9]$ which gives a set of at most $2^{10} + 1$. Filtering the $2^8 - 1$ wrong assignments on the 8 unknowns we can expect 2^{-13} false acceptances in addition to the right one. Simple consistency checks can discard wrong assignments, if any, and recover all k_i's. Clearly, all computations are pretty simple and we only used 2^{10} chosen challenges.

Using the final trick in the Mersenne case we use $d = 6$ and thus 64 chosen challenges to get 64 equations which yield 26 bits each.

Example 5. With $N = 2^{1\,277} - 1$ and the worst case $\ell = r$ the attack works for $b - a \geq 21$ and we can take $d = 19$. We request for 2^{19} chosen challenges. We obtain 2^{19} equations with roughly 1.6 unknowns per equation.

By using the final trick we take $d = 10$. The $T(-\beta N) + \alpha$ part wastes 10 bits from the window and we can expect to have a single unknown per remaining bit so that we can simply read it through the window. Provided that the window has at least 32 bits we expect to read 22 bits in each of the $1\,024$ equations so we can recover all bits.

The narrow window case. When $b - a$ is too small for the previous attack to work, we can still work further on the analysis. To avoid wasting bits on the window, we shall decrease the value for d. Typically, our attack will get less equations and have more unknowns per equation. As a consequence it will list many assignments, including a single correct one. Several directions can be investigated:

- use the highly biased distribution of most significant garbage bits (since α has expected value 0 and standard deviation roughly $2^{\frac{d}{2}}/2\sqrt{3}$);
- use small d and better knapsack solving algorithms;
- analyze these assignments on redundant bits and check for consistency within an equation;
- analyze assignment lists in several equations and try to merge;
- use voting procedures and iterate.

This extension is left to further research.

5 Extending to Non-linear Mappings

In case the mapping L is a (non-linear) *permutation*, we can adapt our attack strategy by choosing the challenges as follow:

- pick d challenges $C_1, \ldots, C_d$.
- compute the chosen challenges by $C^\star(x) = L^{-1}\left(\bigoplus_j x_j L(C_j)\right)$.

By using,

$$c_i^\star(x) = (-1)^{L_i(C^\star(x))} = (-1)^{\bigoplus_j x_j L_i(C_j)} = (-1)^{x \cdot U_i}$$

Equation (3) remains unchanged so that we can still apply all the attacks described through Sections 3 and 4.

More generally, we can extend these attacks to *any* mixing function of form $f(K, C) = g(K) \oplus L(C)$ as long as we can find vector spaces of dimension d in the range of L.

6 Conclusion

One argument for motivating the SQUASH algorithm consisted of playing the "blame game": if anyone can break SQUASH, then the Rabin cryptosystem is the one which should be blamed instead of the SQUASH design. Clearly, our attack demonstrates that this argument is not correct. There are instances of the SQUASH algorithm which can be broken although we still have no clue how to factor integers. Indeed, our method translates into a "known random coins attack" against Rabin-SAEP which leads to a plaintext recovery. Known random coins attacks are not relevant for public-key cryptosystems although they are in the way SQUASH is using it.

It is not clear how and if our attack can be adapted to the final version of SQUASH with non-linear mappings. So, although the "blame game" argument is not valid, the security of SQUASH is still an open problem.

References

1. Boneh, D.: Simplified OAEP for the RSA and rabin functions. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 275–291. Springer, Heidelberg (2001)
2. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
3. Shamir, A.: SQUASH: A new one-way hash function with provable security properties for highly constrained devices such as RFID tags. In: Invited lecture to the RFID Security 2007 Workshop, `http://mailman.few.vu.nl/pipermail/rfidsecuritylist/2007-August/000001.html`
4. Shamir, A.: SQUASH – A new MAC with provable security properties for highly constrained devices such as RFID tags. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 144–157. Springer, Heidelberg (2008)

Practical Chosen Ciphertext Secure Encryption from Factoring

Dennis Hofheinz* and Eike Kiltz**

Cryptology & Information Security Group
CWI Amsterdam, The Netherlands
{hofheinz,kiltz}@cwi.nl

Abstract. We propose a practical public-key encryption scheme whose security against chosen-ciphertext attacks can be reduced in the standard model to the assumption that factoring is intractable.

Keywords: public-key encryption, chosen-ciphertext security, factoring.

1 Introduction

The security of almost any cryptographic primitive (such as public-key encryption or digital signatures) has to rely on the computational hardness of a certain number-theoretic problem. Unfortunately, since there are currently no tools available to rigorously prove lower bounds on the complexity of such problems, one has to base security on (unproven) *cryptographic hardness assumptions*. The only confidence we have in such assumptions is that after a sufficiently large period of time, nobody could successfully refute them. The most established cryptographic hardness assumption is without doubt the so called *factoring assumption* which states that, given the product of two distinct large primes, it is computationally infeasible to reconstruct the primes. Despite of intensive research, no algorithm has been found that can efficiently factor composite numbers.

Main result. In this paper we propose a new public-key encryption scheme that is based on Rabin's trapdoor one-way permutation [35]. We can prove that the security of our scheme against adaptive chosen-ciphertext attacks (CCA security) is equivalent to the factoring assumption. Furthermore, the scheme is practical as its encryption performs only roughly two, and its decryption roughly one modular exponentiation. To the best of our knowledge, this is the first scheme that simultaneously enjoys those two properties.

History. The notion of CCA security is due to Rackoff and Simon [36] and is now widely accepted as the standard security notion for public-key encryption schemes. In contrast to security against passive adversaries (security against

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* Supported by the Dutch Organization for Scientific Research (NWO).
** 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.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 313–332, 2009.

chosen-plaintext attacks aka semantic security), in a chosen-ciphertext attack the adversary plays an active role by obtaining the decryptions of ciphertexts (or even arbitrary bit-strings) of his choosing. The practical significance of such attacks was demonstrated by Bleichenbacher [4] by means of a CCA attack against schemes following the encryption standard PKCS #1.

Historically, the first scheme that was provably secure against CCA attacks is due to Dolev, Dwork, and Naor [17] (building on an earlier result by Naor and Yung [31]). Their generic construction is based on enhanced trapdoor permutations and therefore (using the enhanced trapdoor permutations from [20, App. C]) yields a scheme CCA secure under the factoring assumption. However, in practice these schemes are prohibitively impractical, as they rely on expensive non-interactive zero-knowledge proofs. The first practical schemes provably CCA secure under standard cryptographic hardness assumptions were due to Cramer and Shoup [15,14]. However, their framework of "hash proof systems" inherently relies on *decisional assumptions* such as the assumed hardness of deciding if a given integer has a square root modulo a composite number with unknown factorization (DQR assumption), or of deciding if a given tuple is a Diffie-Hellman tuple or not (DDH assumption). Until today, Cramer and Shoup's framework of hash proof systems (with its variations from [29,19,11,28,24,27]) and the recent concept of lossy trapdoor functions [33] yield the only known CCA secure practical encryption schemes based on an assumption related to factoring: the DQR assumption and Paillier's decisional composite residuosity (DCR) assumption. Currently, no practical scheme is known that is CCA secure solely under the factoring assumption (or even under the potentially stronger RSA assumption).

In general, decisional assumptions are a much stronger class of assumptions than computational assumptions. For example, deciding if a given integer has a modular square root or not may be much easier than actually computing a square root (or, equivalently, factoring the modulus). It is noteworthy that there are known ways to achieve CCA security that do not inherently rely on decisional assumptions (e.g., [9,13,23]). In particular, the first practical encryption scheme CCA secure under the *Computational* Diffie-Hellman (CDH) assumption was only recently proposed by Cash, Kiltz, and Shoup [13] and improved by Hanaoka and Kurosawa [23]. On the other hand, [9] provide a practical encryption scheme CCA secure under the Bilinear Computational Diffie-Hellman (BCDH) assumption.

Random oracle schemes. In a different line of research, Bellare and Rogaway [2,3] presented practical schemes for which they give heuristic proofs of CCA security under standard computational hardness assumptions. Their proofs are in the so-called random oracle model [2] where a hash function is treated as an ideal random function. We stress that although a proof in the random oracle model has a certain value it is still only a heuristic security argument for any implementation of the scheme. In particular, there exist cryptographic schemes that are provably secure in the random oracle model yet that are insecure with *any* possible standard-model instantiation of the hash function [12].

Details of our construction. In 1979 Rabin [35] proposed an encryption scheme based on the "modular squaring" trapdoor permutation whose one-wayness is equivalent to the factoring assumption. A semantically secure variant was later proposed by Goldwasser and Micali [22]. Our construction is based on the latter scheme [22] in its more efficient variant by Blum and Goldwasser [6] (which uses the Blum-Blum-Shub pseudorandom generator [5] to obtain an efficient hard-core function with linear output length). The Blum-Goldwasser scheme can easily be shown insecure against a CCA attack. Our main contribution consists of modifying the Blum-Goldwasser scheme such that it is provably CCA secure under the same hardness assumption yet it retains its high efficiency. Surprisingly, it is sufficient to add one additional group element to the ciphertexts that is then used for a consistency check in the decryption algorithm. For the consistency check itself, we also need to add two group elements to the public key.

Note that Paillier and Villar [32] (building on work of Williams [38]) show that the CCA security of schemes which only include an RSA modulus in the public key cannot be proven (using a black-box reduction) equivalent to factoring. In particular, this applies to the Blum-Goldwasser scheme [6] from which we start, so we have to modify the scheme's public key (and not only the ciphertexts). And indeed, given our modifications, our scheme's CCA security is *equivalent* to the factoring problem.

Proof Details. At a more technical level, the additional group elements in the public key can be set up by a simulator such that it is possible to decrypt (without the knowledge of the scheme's secret key) all consistent ciphertexts, except the ciphertext that is used to challenge the adversary. This "all-but-one" simulation technique can be traced back at least to [30], where it was used in the context of pseudorandom functions.[1] In the encryption context, "all-but-one" simulations have been used in identity-based encryption [8] and were already applied to several encryption schemes in [9,10,13,24,25].

The main novelty is that our proof makes direct use of the fact that the underlying primitive is a trapdoor one-way permutation, rather than the Diffie-Hellman problem. Therefore, the scheme's consistency check can be directly implemented by the simulator *without* having access to some external gap-oracle (as in [9,10,25]) or using other extrinsic rejection techniques (such as "hash proof systems" [15,14], "twinning" [13], or authenticated symmetric encryption [28,24][2]). Thus, our proof technique is fundamentally different from all known approaches

[1] We stress that our use of the term "all-but-one" refers to the ability to generate a secret key that can be used to decrypt all consistent ciphertexts except for an *externally given* ciphertext. This is very different from the techniques of, e.g., [31,16,15]: in these latter frameworks, the first step in the proof consists in *making the challenge ciphertext inconsistent*, and then constructing a secret key that can be used to construct *all* consistent ciphertexts. Hence, "all-but-one" really refers to an "artificially punctured" secret key.

[2] As opposed to generic CCA-secure symmetric encryption, a potentially weaker primitive.

to obtain CCA security. This also includes the recent class of schemes based on lossy trapdoor functions [33].

Efficiency. The resulting encryption scheme (which is actually a key encapsulation mechanism, see [15]) is very efficient: encryption needs roughly two, and decryption roughly one modular exponentiations; the public-key contains the modulus plus two group elements. (The modulus and one element can be viewed as systems parameters shared among all parties). To the best of our knowledge this is much more efficient than all known CCA-secure schemes based on an assumption *related* to factoring, even the ones based on a decisional assumption.

2 Preliminaries

2.1 Notation

We write $[N] = \{1, \ldots, N\}$. For group elements g, h, we denote by $\mathrm{dlog}_g h$ the discrete logarithm of h to the base g, i.e., the smallest $i \geq 0$ with $h = g^i$. A probabilistic polynomial-time (PPT) algorithm is a randomized algorithm which runs in strict polynomial time. If A is a probabilistic algorithm, we write $y \leftarrow A(x)$ to denote that the random variable y is defined as the output of A when run on input x and with fresh random coins. On the other hand, if S is a set, then $s \leftarrow S$ defines s as being uniformly and independently sampled from S. By k we denote the security parameter, which indicates the "amount of security" we desire. Typically, an adversarial advantage should be bounded by 2^{-k}, and a typical value for k is 80.

2.2 Factoring

A prime number P is called a *safe prime* iff $P = 2p + 1$ for a prime p. We assume a PPT algorithm IGen that, on input a security parameter k in unary, generates two random safe primes $P = 2p+1$ and $Q = 2q+1$ with $\mathsf{bitlength}(p) = \mathsf{bitlength}(q) = \ell_{\mathsf{N}}(k)/2 - 1$. We assume that p and q are odd, such that P and Q are congruent 3 modulo 4 and $N = PQ$ is a Blum integer. IGen returns N along with P and Q. Here $\ell_{\mathsf{N}}(k)$ denotes a function that represents, for any given security parameter k, the recommended (bit-)size of the composite modulus N. For the rest of the paper, we assume that N is generated by the factoring instance generator IGen. The set $\mathbb{QR}_N \subseteq \mathbb{Z}_N^*$ of *quadratic residues modulo* N is defined as $\mathbb{QR}_N := \{x \in \mathbb{Z}_N^* : \exists y \in \mathbb{Z}_N^* \text{ with } y^2 = x \bmod N\}$. Since $\mathbb{Z}_N^* \cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_{pq}$, $\mathbb{QR}_N$ is a cyclic group of order pq. Note that this implies that a uniformly chosen element of $\mathbb{QR}_N$ is a generator (of $\mathbb{QR}_N$) with overwhelming probability. Computations in $\mathbb{QR}_N$ are computations modulo N. If it is implied by context, we omit writing explicitly "mod N" for calculations modulo N.

Definition 1 (Factoring assumption). *For an algorithm* F*, we define its* factoring advantage *as*

$$\mathsf{Adv}^{\mathsf{fac}}_{\mathsf{IGen},\mathsf{F}}(k) := \mathsf{Pr}\left[(N, P, Q) \leftarrow \mathsf{IGen}(1^k) \,:\, \mathsf{F}(N) = \{P, Q\}\right].$$

We say that F $(t_{\mathsf{fac}}, \epsilon_{\mathsf{fac}})$-factors composite integers *if* F *runs in time* t_{fac} *and* $\mathsf{Adv}^{\mathsf{fac}}_{\mathsf{IGen},\mathsf{F}}(k) \geq \epsilon(k)$. *The* factoring assumption *(with respect to* IGen*) states that* $\mathsf{Adv}^{\mathsf{fac}}_{\mathsf{IGen},\mathsf{F}}(k)$ *is negligible in* k *for every PPT* F.

The best algorithms currently known for factoring $N = PQ$ of length $\ell_{\mathsf{N}} = \mathsf{bitlength}(N) = \log N$ have (heuristic) running time

$$L_N(1/3, (64/9)^{1/3}) = e^{1.92\ell_{\mathsf{N}}{}^{1/3+o(1)}(\log \ell_{\mathsf{N}})^{2/3}}.$$

Therefore, if we want k bits of security, we need to choose the function $\ell_{\mathsf{N}}(k)$ such that the above term is lower bounded by 2^k. As an example, one commonly uses $\ell_{\mathsf{N}}(80) = 1024$.

2.3 Key Encapsulation Mechanisms

Instead of a public-key encryption scheme we consider the conceptually simpler KEM framework. It is well-known that an $\mathsf{IND\text{-}CCA}$ secure KEM combined with a (one-time-)$\mathsf{IND\text{-}CCA}$ secure symmetric cipher (DEM) yields a $\mathsf{IND\text{-}CCA}$ secure public-key encryption scheme [15]. Efficient *one-time* $\mathsf{IND\text{-}CCA}$ secure DEMs can be constructed even without computational assumptions by using an encrypt-then-MAC paradigm [15] (or, alternatively, using computational assumptions such as strong pseudorandom permutations [34]).

A *key encapsulation mechanism (KEM)* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three PPT algorithms. Via $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, the key generation algorithm produces public/secret keys for security parameter $k \in \mathbb{N}$; via $(K, C) \leftarrow \mathsf{Enc}(pk)$, the encapsulation algorithm creates a symmetric key[3] $K \in \{0,1\}^{\ell_{\mathsf{K}}}$ together with a ciphertext C; via $K \leftarrow \mathsf{Dec}(sk, C)$, the possessor of secret key sk decrypts ciphertext C to get back a key K which is an element in $\{0,1\}^{\ell_{\mathsf{K}}}$ or a special reject symbol $\perp$. For correctness, we require that for all possible $k \in \mathbb{N}$, and all $(K, C) \leftarrow \mathsf{Enc}(pk)$, we have $\Pr[\mathsf{Dec}(sk, C) = K] = 1$, where the probability is taken over the choice of $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, and the coins of all the algorithms in the expression above.

The common requirement for a KEM is indistinguishability against chosen-ciphertext attacks ($\mathsf{IND\text{-}CCA}$) [15], where an adversary is allowed to adaptively query a decapsulation oracle with ciphertexts to obtain the corresponding key. We are using the slightly simpler but equivalent one-phase definition from [26]. Formally:

Definition 2 ($\mathsf{IND\text{-}CCA}$ **security of a KEM).** *Let* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a KEM. For any PPT algorithm* A*, we define the following experiments* $\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM},\mathsf{A}}$ *and* $\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM},\mathsf{A}}$*:*

Experiment $\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM},\mathsf{A}}(k)$	**Experiment** $\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM},\mathsf{A}}(k)$
$(pk, sk) \leftarrow \mathsf{Gen}(1^k)$	$(pk, sk) \leftarrow \mathsf{Gen}(1^k)$
	$R \leftarrow \{0,1\}^{\ell_{\mathsf{K}}}$
$(K^*, C^*) \leftarrow \mathsf{Enc}(pk)$	$(K^*, C^*) \leftarrow \mathsf{Enc}(pk)$
Return $\mathsf{A}^{\mathsf{Dec}(sk,\cdot)}(pk, K^*, C^*)$	*Return* $\mathsf{A}^{\mathsf{Dec}(sk,\cdot)}(pk, R, C^*)$

[3] For simplicity we assume that the KEM's keyspace are bitstrings of length ℓ_{K}.

In the above experiments, the decryption oracle $\mathsf{Dec}(sk,\cdot)$*, when queried with a ciphertext* $C \neq C^*$*, returns* $K \leftarrow \mathsf{Dec}(sk, C)$*. (*$\mathsf{Dec}(sk,\cdot)$ *ignores queries* $C = C^*$*.) We define* A*'s advantage in breaking* KEM*'s* $\mathsf{IND\text{-}CCA}$ *security as*

$$\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM},\mathsf{A}}(k) := \frac{1}{2}\left|\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM},\mathsf{A}}(k)=1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM},\mathsf{A}}(k)=1\right]\right|.$$

A $(t_{\mathsf{KEM}}, \epsilon_{\mathsf{KEM}})$-breaks KEM's $\mathsf{IND\text{-}CCA}$ security (short: A $(t_{\mathsf{KEM}}, \epsilon_{\mathsf{KEM}})$-breaks KEM) *if* A *runs in time at most* $t_{\mathsf{KEM}} = t_{\mathsf{KEM}}(k)$ *and we have* $\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM},\mathsf{A}}(k) \geq \epsilon_{\mathsf{KEM}}(k)$*. We say that* KEM has indistinguishable ciphertexts under chosen-ciphertext attacks (short: KEM is $\mathsf{IND\text{-}CCA}$ secure) *if for all PPT* A*, the function* $\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM},\mathsf{A}}(k)$ *is negligible in* k*.*

2.4 Target-Collision Resistant Hashing

Informally, we say that a function $\mathsf{T} : X \to Y$ is a target-collision resistant (TCR) hash function (aka universal one-way hash function [31]), if, given a random preimage $x \in X$, it is hard to find $x' \neq x$ with $\mathsf{T}(x') = \mathsf{T}(x)$.

Definition 3 (TCR hash function). *Let* $\mathsf{T} : X \to Y$ *be a function. For an algorithm* B*, define*

$$\mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k) := \mathsf{Pr}\left[x \leftarrow X, x' \leftarrow \mathsf{B}(x) \,:\, x' \neq x \wedge \mathsf{T}(x') = \mathsf{T}(x)\right].$$

We say that B $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$-breaks T's TCR property (short: B $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$-breaks T) *iff* B*'s running time is at most* $t_{\mathsf{T}}(k)$ *and* $\mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k) \geq \epsilon_{\mathsf{T}}(k)$*. We say that* T *is* target-collision resistant *if for all PPT* B*, the function* $\mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k)$ *is negligible in* k*.*

3 Chosen-Ciphertext Security from Factoring

3.1 The Scheme

In this section, we will present our KEM construction. We will make use of two building blocks: a target collision-resistant hash function, and the Blum-Blum-Shub (BBS) pseudorandom number generator [5].

Concretely, for a product $N = PQ$ of two primes P, Q and $u \in \mathbb{Z}_N$, we establish the following notation: $|u|$ denotes the absolute value of u and $\mathsf{LSB}_N(u) = u \bmod 2$ the least significant bit of u, where in both cases u is interpreted as a signed integer with $-(N-1)/2 \leq u \leq (N-1)/2$. Furthermore, let

$$\mathsf{BBS}_N(u) = \left(\mathsf{LSB}_N(u), \mathsf{LSB}_N(u^2) \ldots, \mathsf{LSB}_N(u^{2^{\ell_K - 1}})\right) \in \{0,1\}^{\ell_K}$$

denote the BBS generator applied to u and modulo N.[4]

[4] For efficiency, and at the price of a worse reduction, one can even simultaneously extract $\lceil \log_2 \log_2 N \rceil$ bits of each u^{2^i} instead of only the least significant bit [1]. However, our analysis treats the original BBS generator for simplicity.

Furthermore, for N as above, let $\mathsf{T} : \mathbb{Z}_N \to \{1, \ldots, 2^{\ell_\mathsf{T}} - 1\}$ be a target-collision resistant hash function.

The scheme. We are ready to define the following key encapsulation mechanism $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

Key generation. $\mathsf{Gen}(1^k)$ chooses uniformly at random

- a modulus $N = PQ = (2p+1)(2q+1)$ (using $\mathsf{IGen}(1^k)$, cf. Section 2.2),
- a quadratic residue $g \in \mathbb{QR}_N$,
- an exponent $\alpha \in [(N-1)/4]$,

Gen then sets $X = g^{\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$ and outputs a public key pk and a secret key sk, where

$$pk = (N, g, X) \qquad\qquad sk = (N, g, \alpha).$$

Encapsulation. $\mathsf{Enc}(pk)$ chooses uniformly $r \in [(N-1)/4]$, sets

$$R = g^{r 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \qquad t = \mathsf{T}(R) \in \{1, \ldots, 2^{\ell_\mathsf{T}} - 1\} \qquad S = \left| \left(g^t X\right)^r \right|$$

and outputs the key $K = \mathsf{BBS}_N(g^{r 2^{\ell_\mathsf{T}}}) \in \{0,1\}^{\ell_\mathsf{K}}$ and the ciphertext $C = (R, S) \in \mathbb{QR}_N \times (\mathbb{Z}_N^* \cap [(N-1)/2])$.

Decapsulation. $\mathsf{Dec}(sk, (R, S))$ verifies that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$ and rejects if not. Then, Dec computes $t = \mathsf{T}(R) \in \{1, \ldots, 2^{\ell_\mathsf{T}} - 1\}$, checks whether

$$\left(S^2\right)^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \stackrel{?}{=} \left(R^2\right)^{t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \tag{1}$$

holds, and rejects if not. If (1) holds, Dec computes $a, b, c \in \mathbb{Z}$ such that

$$2^c = \gcd(t, 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}) = at + b2^{\ell_\mathsf{K}+\ell_\mathsf{T}}. \tag{2}$$

Note that $c < \ell_\mathsf{T}$ since $0 < t < 2^{\ell_\mathsf{T}}$. Then, Dec derives

$$T = \left(\left(S^2\right)^a \cdot \left(R^2\right)^{b-a\alpha}\right)^{2^{\ell_\mathsf{T}-c-1}} \tag{3}$$

and from this $K = \mathsf{BBS}_N(T) \in \{0,1\}^{\ell_\mathsf{K}}$, which is the output.

We remark that decapsulation (or, rather, generation of the secret keys) does not require knowledge about the factorization of N. Indeed, the modulus N as well as the generator g can be viewed as global system parameters shared by many parties. Then pk only contains the value $X \in \mathbb{QR}_N$ and sk only contains $\alpha \in [(N-1)/4]$.

Our scheme uses an RSA modulus N that consists of safe primes. In Section 5 we show how to avoid this assumption and allow N to be an arbitrary Blum integer.

Correctness. The correctness of the scheme might not be obvious, so we prove it here. Fix a public key pk and a secret key sk as produced by $\mathsf{Gen}(1^k)$, and assume that (R, S) is a ciphertext for a key K as generated by $\mathsf{Enc}(pk)$. We

have to show that $\mathsf{Dec}(sk,(R,S))$ outputs K. First, it is clear that $(R,S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$. Also,

$$\left(S^2\right)^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} = \left(\left|\left(g^t X\right)^r\right|^2\right)^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} = g^{2(t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}})r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \overset{(*)}{=} \left(R^2\right)^{t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$$

(where $(*)$ uses $R = g^{r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$), so (1) holds. Hence, (R,S) is not rejected by Dec. Now (1) implies

$$S^2 = \left(R^2\right)^{\frac{t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}} = \left(R^2\right)^{\frac{t}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}+\alpha}, \tag{4}$$

where the division in the exponent is computed modulo $pq = |\mathbb{QR}_N|$. (Note that while S may or may not be a quadratic residue, S^2 certainly is.) This gives

$$\begin{aligned} T &\overset{(3)}{=} \left(\left(S^2\right)^a \cdot \left(R^2\right)^{b-a\alpha}\right)^{2^{\ell_\mathsf{T}-c-1}} = \left(\left(S^2 \cdot \left(R^2\right)^{-\alpha}\right)^a \cdot \left(R^2\right)^b\right)^{2^{\ell_\mathsf{T}-c-1}} \\ &\overset{(4)}{=} \left(\left(\left(R^2\right)^{\frac{t}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}}\right)^a \cdot \left(R^2\right)^b\right)^{2^{\ell_\mathsf{T}-c-1}} = \left(\left(R^2\right)^{\frac{at+b2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}}\right)^{2^{\ell_\mathsf{T}-c-1}} \\ &\overset{(2)}{=} \left(R^2\right)^{\frac{2^c}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}\cdot 2^{\ell_\mathsf{T}-c-1}} = \left(R^2\right)^{\frac{1}{2^{\ell_\mathsf{K}+1}}} \overset{(*)}{=} g^{r2^{\ell_\mathsf{T}}}, \end{aligned} \tag{5}$$

where, again, $(*)$ uses $R = g^{r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$. But (5) shows that Dec outputs $\mathsf{BBS}_N(T) = \mathsf{BBS}_N(g^{r2^{\ell_\mathsf{T}}}) = K$ as desired.

Theorem 1 (IND-CCA **security of** KEM). *Assume* T *is a target collision resistant hash function and the factoring assumption holds. Then* KEM *is* IND-CCA *secure in the sense of Definition 2.*

The proof of Theorem 1 will be given in Section 4.

Efficiency. We claim that, with some trivial optimizations, encapsulation uses roughly two exponentiations, and decapsulation roughly one exponentiation. Namely, encapsulation can first compute $A = g^r$ and $B = X^r$, which are two full exponentiations. Then, the remaining computations require only multiplications or exponentiations with very small exponents: $K = \mathsf{BBS}_N(A^{2^{\ell_\mathsf{T}}})$, $R = A^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$, and $S = A^t B$. (In fact, R is a by-product of computing K.) Similarly, decapsulation can first compute $D = R^\alpha / S$, which requires one full exponentiation. From D, (1) can be checked with $D^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}+1}} \overset{?}{=} R^{2t}$, which requires only two exponentiations with very small exponents. The key K can then be computed as $\mathsf{BBS}_N(T)$ for $T = (R^b D^{-a})^{2^{\ell_\mathsf{T}-c}}$, which requires three exponentiations with small exponents (note that the bit-length of a and b is at most $\ell_\mathsf{K} + \ell_\mathsf{T}$).

For concreteness let us assume that one regular exponentiation with an exponent of length ℓ requires $1.5 \cdot \ell$ modular multiplications and that one squaring takes the same time as one multiplication. Let us further assume that $\ell_\mathsf{N} := \mathsf{bitlength}(N) = 1024$ and $\ell_\mathsf{K} = \ell_\mathsf{T} = 80$. Then encapsulation requires $3\ell_\mathsf{N} + \ell_\mathsf{K} + 2.5\ell_\mathsf{T} = 3352$ multiplications; decapsulation requires $1.5\ell_\mathsf{N} + 4\ell_\mathsf{K} + 6.5\ell_\mathsf{T} = 2376$ multiplications. In Appendix A we also propose a variant of our

scheme that has slightly more efficient decapsulation but suffers from a comparatively large public key size.

We remark that, by adding the prime factors P and Q to the secret-key, we can further improve the scheme's efficiency. For example, using Chinese Remaindering will speed up decapsulation by a factor between 3 and 4.

4 Proof of Security

We split up the proof of Theorem 1 into two parts:

- We first recall that the BBS generator is pseudorandom if factoring Blum integers is hard. This holds even if the modulus N and the 2^{ℓ_K}-th power $u^{2^{\ell_K}}$ of the BBS seed u are published, as is the case in our KEM. (Theorem 2.)
- We then prove that KEM is IND-CCA secure under the assumption that the BBS generator is pseudorandom and the employed hash function is target-collision resistant. This reduction is the heart of our proof. (Theorem 3.)

Combining both parts yields Theorem 1.

We start by recalling that the BBS generator is pseudorandom, in the following sense.

Definition 4 (PRNG experiment for BBS generator). *For an algorithm* D*, define*

$$\mathsf{Adv}_{\mathsf{D}}^{\mathsf{BBS}}(k) = \mathsf{Pr}\left[\mathsf{D}(N, z, \mathsf{BBS}_N(u)) = 1\right] - \mathsf{Pr}\left[\mathsf{D}(N, z, U_{\{0,1\}^{\ell_K}}) = 1\right],$$

where

- $N \in \mathbb{N}$ *is distributed as* $\mathsf{IGen}(1^k)$,
- $u \in \mathbb{QR}_N$ *is uniformly chosen, and* $z = u^{2^{\ell_K}}$,
- $U_{\{0,1\}^{\ell_K}} \in \{0,1\}^{\ell_K}$ *is independently and uniformly chosen.*

We say that D (t, ϵ)-breaks BBS *if* D*'s running time is at most* $t = t(k)$ *and* $\mathsf{Adv}_{\mathsf{D}}^{\mathsf{BBS}}(k) \geq \epsilon = \epsilon(k)$.

Concretely, any BBS-distinguisher can be used to factor Blum integers.

Theorem 2 (BBS-distinguisher $\Rightarrow$ factoring algorithm [7,5,1,18]). *For every algorithm* D *that* $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$*-breaks* BBS*, there is an algorithm* F *that* $(t_{\mathsf{fac}}, \epsilon_{\mathsf{fac}})$*-factors Blum integers, where*

$$t_{\mathsf{fac}} \approx k^4 t_{\mathsf{BBS}}/\epsilon_{\mathsf{BBS}}^2 \qquad\qquad \epsilon_{\mathsf{fac}} = \epsilon_{\mathsf{BBS}}/\ell_{\mathsf{K}}.$$

Proof. Let D be an algorithm that $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$-breaks BBS. [7] show that D gives rise to an algorithm D′ that $(t_{\mathsf{LSB}}, \epsilon_{\mathsf{LSB}})$-distinguishes tuples $(N, u^2, \mathsf{LSB}(u))$ from tuples $(N, u^2, U_{\{0,1\}})$, where $u \in \mathbb{QR}_N$ and $U_{\{0,1\}} \in \{0,1\}$ are uniformly chosen, $t_{\mathsf{LSB}} \approx t_{\mathsf{BBS}}$, and $\epsilon_{\mathsf{LSB}} = \epsilon_{\mathsf{BBS}}/\ell_{\mathsf{K}}$. Building on [1], [18] show how to transform D′ into an algorithm F that $(t_{\mathsf{fac}}, \epsilon_{\mathsf{fac}})$-factors Blum integers, where $t_{\mathsf{fac}} \approx k^2 t_{\mathsf{LSB}}/\epsilon_{\mathsf{LSB}}^2 \approx k^4 t_{\mathsf{BBS}}/\epsilon_{\mathsf{BBS}}^2$ and $\epsilon_{\mathsf{fac}} = \epsilon_{\mathsf{LSB}} = \epsilon_{\mathsf{BBS}}/\ell_{\mathsf{K}}$. (We use the interpretation [30, Theorem 6.1] of the results from [18] here.) The claim follows.

The following theorem contains the heart of our proof, namely, a simulation that shows that any successful IND-CCA adversary on KEM implies a successful BBS-distinguisher (and hence, using Theorem 2, can be used to factor Blum integers).

Theorem 3 (IND-CCA adversary ⇒ BBS-distinguisher). *For every adversary* A *that* $(t_{\mathsf{KEM}}, \epsilon_{\mathsf{KEM}})$*-breaks* KEM*'s* IND-CCA *property, there exists an algorithm* D *that* $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$ *breaks* BBS *and an adversary* B *that* $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$*-breaks* T*, such that*

$$t_{\mathsf{BBS}} \approx t_{\mathsf{T}} \approx t_{\mathsf{KEM}} \qquad \epsilon_{\mathsf{BBS}} + \epsilon_{\mathsf{T}} + 2^{-k+3} \geq \epsilon_{\mathsf{KEM}}.$$

Proof. **Setting up the variables for simulation.** Assume an adversary A on KEM's IND-CCA security. We define a BBS-distinguisher D, which acts on input (N, z, V) as follows. D first uniformly selects a quadratic residue $g \in \mathbb{QR}_N$, as well as exponent $\beta \in [(N-1)/4]$, and sets

$$R^* = z \qquad t^* = \mathsf{T}(R^*) \in \{1, \ldots, 2^{\ell_{\mathsf{T}}} - 1\} \qquad X = g^{\beta 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}} - t^*}.$$

The public key used in the simulation is $pk = (N, g, X)$. It will be convenient to write $X = g^{\alpha 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}$ as in Gen, for $\alpha = \beta - t^*/2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}$ unknown to D. (Here and in the following, a division of exponents is computed modulo pq, the order of $\mathbb{QR}_N$.) Furthermore, in the following, we will silently assume that g generates $\mathbb{QR}_N$, which is very likely, but not guaranteed. A rigorous justification that takes into account error probabilities follows below.

Preparation of challenge ciphertext and key. To complete the definition of the challenge ciphertext $C^* = (R^*, S^*)$, write $R^* = g^{2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}} r^*}$. Since we assumed that g is a generator, this is possible, but of course r^* is unknown. D defines

$$S^* = \left|R^{*\beta}\right| \quad \left(= \left|g^{r^*\beta 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}\right| = \left|\left(g^{t^*} X\right)^{r^*}\right|\right) \tag{6}$$

as Enc would have computed. The (real) corresponding key K^* is defined as

$$K^* = \mathsf{BBS}_N\left(g^{2^{\ell_{\mathsf{T}}} r^*}\right) = \mathsf{BBS}_N\left(R^{* \frac{1}{2^{\ell_{\mathsf{K}}}}}\right) = \mathsf{BBS}_N\left(z^{\frac{1}{2^{\ell_{\mathsf{K}}}}}\right) = \mathsf{BBS}_N(u)\ . \tag{7}$$

D then invokes A with public key $pk = (N, g, X)$, challenge ciphertext $C^* = (R^*, S^*)$, and challenge key V. Note that V is either the real challenge key $\mathsf{BBS}_N(u)$, or it is a uniform string.

On the distribution of simulated public key and challenge ciphertext. We claim that the distribution of public key pk and challenge ciphertext C^* is almost identical in simulation and IND-CCA experiment. Concretely, we postpone the straightforward but somewhat tedious proof of the following lemma until after the description of our simulation.

Lemma 1. *There exists an event* $\mathsf{bad}_{\mathsf{key}}$ *such that, conditioned on* $\neg\mathsf{bad}_{\mathsf{key}}$*, public key pk and challenge ciphertext* C^* *are identically distributed in simulation and* IND-CCA *experiment. Also,* $\neg\mathsf{bad}_{\mathsf{key}}$ *implies that g is a generator. We have*

$$\mathsf{Pr}\left[\mathsf{bad}_{\mathsf{key}}\right] \leq 2^{-k+3} \tag{8}$$

both in the simulation and in the IND-CCA *experiment.*

Thus, conditioned on $\neg\mathsf{bad}_{\mathsf{key}}$, D perfectly simulates A's input as in the real IND-CCA experiment if $V = \mathsf{BBS}_N(u) = \mathsf{BBS}_N(z^{1/2^{\ell_\mathsf{K}}})$, and as in the ideal IND-CCA experiment if V is random.

How to handle A's decryption queries. It remains to describe how D handles decryption queries of A as in the IND-CCA experiment. So say that A submits a ciphertext (R, S) for decryption. We may assume that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$. Let $t = \mathsf{T}(R) \in \{1, \ldots, 2^{\ell_\mathsf{T}} - 1\}$. We call a ciphertext *consistent* iff the original decryption algorithm would not have rejected it. Hence, by (1), a ciphertext is consistent iff

$$\left(S^2\right)^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \stackrel{?}{=} \left(R^2\right)^{t-t^*+\beta 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \quad \left(= \left(R^2\right)^{t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}\right) . \tag{9}$$

By our setup of variables, D can check (9) by itself, and hence detect and reject inconsistent ciphertexts.

How to decrypt consistent ciphertexts. Now assume that C is consistent and $t \neq t^*$. Then, (4) and (5) follow (except for the deduction $(*)$) just as in the correctness proof, and we get

$$T = \left(R^2\right)^{\frac{1}{2^{\ell_\mathsf{K}+1}}} \tag{10}$$

for the raw key T that would have been computed by Dec. We will now show how D can compute T. Namely, D computes $a', b', c' \in \mathbb{Z}$ such that

$$2^{c'} = \gcd(t - t^*, 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}) = a'(t - t^*) + b' 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}. \tag{11}$$

Since $1 \leq t, t^* < 2^{\ell_\mathsf{T}}$ and $t \neq t^*$, we have $c' < \ell_\mathsf{T}$. Similarly to (4) and (5), we obtain

$$S^2 = \left(R^2\right)^{\frac{t-t^*}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}+\beta}, \tag{12}$$

from (9), and from this

$$\begin{aligned}
\left(\left(S^2\right)^{a'} \cdot \left(R^2\right)^{b'-a'\beta}\right)^{2^{\ell_\mathsf{T}-c'-1}} &= \left(\left(S^2 \cdot \left(R^2\right)^{-\beta}\right)^{a'} \cdot \left(R^2\right)^{b'}\right)^{2^{\ell_\mathsf{T}-c'-1}} \\
\stackrel{(12)}{=} \left(\left(\left(R^2\right)^{\frac{t-t^*}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}}\right)^{a'} \cdot \left(R^2\right)^{b'}\right)^{2^{\ell_\mathsf{T}-c'-1}} &= \left(\left(R^2\right)^{\frac{a'(t-t^*)+b'2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}}\right)^{2^{\ell_\mathsf{T}-c'-1}} \\
&\stackrel{(11)}{=} \left(R^2\right)^{\frac{2^{c'}}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \cdot 2^{\ell_\mathsf{T}-c'-1}} = \left(R^2\right)^{\frac{1}{2^{\ell_\mathsf{K}+1}}} \stackrel{(10)}{=} T.
\end{aligned} \tag{13}$$

Note that from T, the final decryption key can be computed as $K = \mathsf{BBS}_N(T)$. Hence, using (13), D can correctly decrypt every consistent ciphertext with $t \neq t^*$.

The case $t = t^*$. So let us turn to the case that $t = t^*$ and the ciphertext is consistent. Then, if $R = R^*$ holds, we have

$$S^2 \overset{(9)}{=} \left(R^2\right)^{\frac{t-t^*}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}+\beta} \overset{(*)}{=} \left(R^{*2}\right)^{\beta} = S^{*2} \tag{14}$$

where in $(*)$ we use $R = R^*$ and $t = t^*$. Furthermore, $(R, S) \neq (R^*, S^*)$ implies $|S| = S \neq S^* = |S^*|$, so that $S \neq \pm S^*$ and $(S + S^*)(S - S^*) = S^2 - S^{*2} \overset{(14)}{=} 0 \bmod N$ yields a non-trivial factorization of N. Hence, D can efficiently factor N to solve its own input challenge (N, z, L) directly whenever $R = R^*$ and (R, S) is consistent.

On the other hand, if $\mathsf{T}(R) = t = t^* = \mathsf{T}(R^*)$ and $R \neq R^*$, then A has broken the target-collision resistance of T. Formally, let $\mathsf{bad}_{\mathsf{TCR}}$ denote the event that $t = t^*$ and $R \neq R^*$. If $\mathsf{bad}_{\mathsf{TCR}}$ occurs, D can safely give up, since

$$\mathsf{Pr}\left[\mathsf{bad}_{\mathsf{TCR}}\right] \leq \mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k) \tag{15}$$

for a suitable PPT adversary B on T that simulates D and A.

Summary of the decryption procedure. We summarize the decryption cases:

- inconsistent (R, S) (consistency check (9)$\Leftrightarrow$(1) not passed): reject,
- consistent (R, S) and $t \neq t^*$: decrypt using (13),
- consistent (R, S), $t = t^*$, and $R = R^*$: factor N (using $S \neq \pm S^*$ and $S^2 = S^{*2}$ by (14)),
- consistent (R, S), $t = t^*$, and $R \neq R^*$: give up simulation (A has found a T-collision).

Hence, also decryption is faithfully simulated unless $\mathsf{bad}_{\mathsf{TCR}}$ occurs.

Finishing the proof. We conclude that, unless $\mathsf{bad}_{\mathsf{TCR}}$ or $\mathsf{bad}_{\mathsf{key}}$ occurs, D perfectly simulates the real IND-CCA experiment upon input $V = \mathsf{BBS}_N(u)$, and the ideal IND-CCA experiment if V is random. If we let D output whatever the simulated experiment outputs, we obtain:

$$\begin{aligned}
&\left|\mathsf{Pr}\left[\mathsf{D}(N, z, \mathsf{BBS}_N(u)) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM},\mathsf{A}}(k) = 1\right]\right| \leq \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{TCR}}\right] + \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{key}}\right] \\
&\left|\mathsf{Pr}\left[\mathsf{D}(N, z, U_{\{0,1\}^{\ell_\mathsf{K}}}) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM},\mathsf{A}}(k) = 1\right]\right| \leq \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{TCR}}\right] + \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{key}}\right].
\end{aligned} \tag{16}$$

Using (8) and (15), Theorem 3 follows from (16).

It remains to prove Lemma 1.

Proof of Lemma 1. Observe that pk and C^* are distributed slightly differently in the IND-CCA experiment (i.e., as generated by Gen and Enc) and in the simulation:

- $R^* = g^{r^*}$ for uniform (hidden) $r^* \in [(N-1)/4]$ in the experiment, while $R^* \in \mathbb{QR}_N$ is a uniform group element in the simulation.
- $X = g^{\alpha 2^{\ell_K+\ell_T}}$ for uniform (hidden) $\alpha \in [(N-1)/4]$ in the experiment, while $X = g^{\beta 2^{\ell_K+\ell_T}-t^*}$ for uniform (hidden) $\beta \in [(N-1)/4]$ in the simulation.

However, conditioned on the following event $\mathsf{good}_{\mathsf{key}}$:

(in the experiment:) g is a generator, and $r^*, \alpha \leq |\mathbb{QR}_N|$,

(in the simulation:) g is a generator, and $\beta \leq |\mathbb{QR}_N|$,

pk and C^* are distributed identically in experiment and simulation: $\mathsf{good}_{\mathsf{key}}$ implies that N, g, X, and R^* are uniformly and independently chosen over their respective domains, and S^* follows deterministically from pk and R^* according to (7). Hence we only need to bound the probability of $\mathsf{bad}_{\mathsf{key}} := \neg\mathsf{good}_{\mathsf{key}}$. Since $|\mathbb{QR}_N| = pq$ and we assumed that p and q are $n/2$-bit primes, a uniform $\mathbb{QR}_N$-element is a generator except with probability $(p+q-1)/pq \leq 2^{-n/2+2}$. Furthermore, $(N-1)/4$ is a close approximation of the group order $|\mathbb{QR}_N| = pq = (N-1)/4 - (p+q)/2$, so that, e.g., $r^* \leq |\mathbb{QR}_N|$ except with probability $2(p+q)/(N-1) \leq 2^{-n/2+1}$. Hence,

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{bad}_{\mathsf{key}}\right] &\leq \max\left\{2^{-n/2+2} + 2\cdot 2^{-n/2+1}, 2^{-n/2+2} + 2^{-n/2+1}\right\} \\ &= 2^{-n/2+3} \overset{n/2\geq k}{\leq} 2^{-k+3}\end{aligned}$$

both in the experiment and in the simulation.

5 Avoiding Safe Primes

In our KEM, we assume that $N = PQ$ is composed of two safe primes (i.e., primes of the form $P = 2p+1$ for prime p). We can drop this assumption and allow arbitrary Blum integers N, if we employ a Goldreich-Levin [21] based pseudorandom generator instead of the Blum-Blum-Shub generator. Namely, all we actually need to prove that KEM is $\mathsf{IND\text{-}CCA}$ is that

$$\left(N, g, g^{r2^{\ell_K+\ell_T}}, \mathsf{Ext}_{pk}(g^{r2^{\ell_T}})\right) \overset{c}{\approx} \left(N, g, g^{r2^{\ell_K+\ell_T}}, U_{\{0,1\}^{\ell_K}}\right), \tag{17}$$

where $\overset{c}{\approx}$ denotes computational indistinguishability, N is a Blum integer, $g \in \mathbb{QR}_N$, $r \in [N/4]$, and $U_{\{0,1\}^{\ell_K}} \in \{0,1\}^{\ell_K}$ are uniform, and Ext is a suitable randomness extractor. In our original description of KEM, we have $\mathsf{Ext}_{pk}(u) = \mathsf{BBS}_N(u)$. In that case, we only know that the hardness of factoring N implies (17) if $u = g^{r2^{\ell_T}}$ is a *uniform* element of $\mathbb{QR}_N$ (which is the case when $N = PQ$ for safe primes P, Q, since then g is a generator with high probability). But if g is not a generator at least with high probability, then u may not be uniformly distributed.

Now suppose we set

$$\mathsf{Ext}_{pk}(u) = \left(\mathsf{GL}_s(u), \mathsf{GL}_s(u^2), \ldots, \mathsf{GL}_s(u^{2^{\ell_K-1}})\right) \in \{0,1\}^{\ell_K}$$

for the Goldreich-Levin predicate GL_s that maps u to the bitwise inner product of s and u. Then a hybrid argument and the hard-core property of GL_s show that (17) is implied by the hardness of computing u with $u^2 = v \mod N$ from (N, g, v) (with $v = g^r$). But any algorithm B that computes such a u from (N, g, v) can be used to factor N. Namely, given N, choose uniformly $h \in \mathbb{Z}_N$ and $\tilde{r} \in [N/4]$, and set $g = h^2$ and $v = g^{2\tilde{r}+1}$. (Observe that v is almost uniformly distributed over $\langle g \rangle$, since N is a Blum integer.) Then, invoke $B(N, g, v)$ to obtain a square root u of v. We can then compute a square root of g as $\tilde{h} = u^a g^b$ (for $a, b \in \mathbb{Z}$ with $a(2\tilde{r}+1) + 2b = \gcd(2\tilde{r}+1, 2) = 1$). With probability $1/2$, then $\gcd(h - \tilde{h}, N)$ yields a non-trivial factor of N. Hence (17) is implied by the hardness of factoring arbitrary Blum integers, and our KEM (instantiated with the Goldreich-Levin predicate) is $\mathsf{IND\text{-}CCA}$ secure. The price to pay is that we need to place a seed $s \in \{0,1\}^{\ell_N}$ for the Goldreich-Levin hard-core function in the public key. (However, note that s can be made a global system parameter, like N and g.)

Acknowledgements

We are grateful to Victor Shoup, who generously allowed us to use his observations on how to compress the public key from $O(\ell_{\mathsf{T}})$ down to two group elements, and on how to get rid of the assumption that P and Q are safe primes. We would also like to thank Ronald Cramer and Ivan Damgård for interesting discussions.

References

1. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.-P.: RSA and Rabin functions: Certain parts are as hard as the whole. SIAM Journal on Computing 17(2), 194–209 (1988)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
3. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
4. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
5. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM Journal on Computing 15(2), 364–383 (1986)
6. Blum, M., Goldwasser, S.: An efficient probabilistic public-key encryption scheme which hides all partial information. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 289–302. Springer, Heidelberg (1985)
7. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudorandom bits. SIAM Journal on Computing 13(4), 850–864 (1984)
8. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

9. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM Journal on Computing 36(5), 915–942 (2006)
10. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS 2005, pp. 320–329. ACM Press, New York (2005)
11. Camenisch, J.L., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
12. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Journal of the ACM 51(4), 557–594 (2004)
13. Cash, D.M., Kiltz, E., Shoup, V.: The twin diffie-hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
14. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
15. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
16. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: 23rd ACM STOC, pp. 542–552. ACM Press, New York (1991)
17. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
18. Fischlin, R., Schnorr, C.-P.: Stronger security proofs for RSA and Rabin bits. Journal of Cryptology 13(2), 221–244 (2000)
19. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. ACM Transactions on Information and System Security 9(2), 181–234 (2006)
20. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
21. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC, pp. 25–32. ACM Press, New York (1989)
22. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
23. Hanaoka, G., Kurosawa, K.: Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, pp. 308–325. Springer, Heidelberg (2008)
24. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
25. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
26. Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
27. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS. Springer, Heidelberg (2009)
28. Kurosawa, K., Desmedt, Y.G.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)

29. Lucks, S.: A variant of the cramer-shoup cryptosystem for groups of unknown order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 27–45. Springer, Heidelberg (2002)
30. Naor, M., Reingold, O., Rosen, A.: Pseudo-random functions and factoring. SIAM Journal on Computing 31(5), 1383–1404 (2002)
31. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. ACM Press, New York (1990)
32. Paillier, P., Villar, J.L.: Trading one-wayness against chosen-ciphertext security in factoring-based encryption. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 252–266. Springer, Heidelberg (2006)
33. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 187–196. ACM Press, New York (2008)
34. Phan, D.H., Pointcheval, D.: About the security of ciphers (Semantic security and pseudo-random permutations). In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 182–197. Springer, Heidelberg (2004)
35. Rabin, M.O.: Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology (January 1979)
36. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
37. Reyzin, L., Reyzin, N.: Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 144–154. Springer, Heidelberg (2002)
38. Williams, H.C.: A modification of the RSA public-key encryption procedure. IEEE Transactions on Information Theory 26(6), 726–729 (1980)

A A Variant of Our Scheme

We now propose a variant of our scheme that has slightly more efficient decapsulation but suffers from a comparatively large public key size.

Let $\mathsf{T} : \mathbb{Z}_N \to \{0,1\}^{\ell_\mathsf{T}}$ be a target-collision resistant hash function. Then, define the following key encapsulation mechanism $\mathsf{KEM}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$:

Key generation. $\mathsf{Gen}'(1^k)$ chooses uniformly at random

- a modulus $N = PQ = (2p+1)(2q+1)$ (using $\mathsf{IGen}(1^k)$, cf. Section 2.2),
- a quadratic residue $h \in \mathbb{QR}_N$,
- exponents $\alpha_{i,j} \in [(N-1)/4]$ for $i \in [\ell_\mathsf{T}]$ and $j \in \{0,1\}$,

Gen' then gets $g = h^2$ and

$$X_{1,j} = g^{\alpha_{1,j} 2^{\ell_\mathsf{K}}} \cdot h \qquad j \in \{0,1\}$$
$$X_{i,j} = g^{\alpha_{i,j} 2^{\ell_\mathsf{K}}} \qquad i = 2, \ldots, \ell_\mathsf{T}, \quad j \in \{0,1\}.$$

Gen' finally outputs a public key pk and a secret key sk, where

$$pk = (N, g, (X_{i,j})_{i \in [\ell_\mathsf{T}], j \in \{0,1\}}) \qquad sk = (N, (\alpha_{i,j})_{i \in [\ell_\mathsf{T}], j \in \{0,1\}}).$$

Encapsulation. $\mathsf{Enc}'(pk)$ chooses uniformly $r \in [(N-1)/4]$, sets

$$R = g^{2^{\ell_\mathsf{K}} r} \qquad t = (t_1, \ldots, t_{\ell_\mathsf{T}}) = \mathsf{T}(R) \in \{0,1\}^{\ell_\mathsf{T}} \qquad S = \left| \left(\prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i} \right)^r \right|$$

and outputs the key $K = \mathsf{BBS}_N(g^r) \in \{0,1\}^{\ell_\mathsf{K}}$ and the ciphertext $C = (R, S) \in \mathbb{QR}_N \times (\mathbb{Z}_N^* \cap [(N-1)/2])$.

Decapsulation. $\mathsf{Dec}'(sk, (R, S))$ verifies that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$ and rejects if not. Then, Dec' computes $t = (t_1, \ldots, t_{\ell_\mathsf{T}}) = \mathsf{T}(R)$ and checks

$$\left(S^2\right)^{2^{\ell_\mathsf{K}+1}} \stackrel{?}{=} \left(R^2\right)^{1+2^{\ell_\mathsf{K}+1} \sum_{i=1}^{\ell_\mathsf{T}} \alpha_{i,t_i}} \qquad \left(= \left(R^2\right)^{2\mathrm{dlog}_g \prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i}} \right). \tag{18}$$

If (18) does not hold, Dec' rejects. Otherwise, Dec' computes

$$T = S^2 \left(R^2\right)^{-\sum_{i=1}^{\ell_\mathsf{T}} \alpha_{i,t_i}} \qquad \left(\stackrel{(18)}{=} \left(R^2\right)^{\frac{1}{2^{\ell_\mathsf{K}+1}}} \right). \tag{19}$$

and outputs $K = \mathsf{BBS}_N(T) = \mathsf{BBS}_N((R^2)^{\frac{1}{2^{\ell_\mathsf{K}+1}}})$.

Correctness. The scheme enjoys correctness, since an honestly generated ciphertext (R, S) fulfils

$$\begin{aligned} \left(S^2\right)^{2^{\ell_\mathsf{K}+1}} &= \left(\left| \left(\prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i} \right)^r \right|^2 \right)^{2^{\ell_\mathsf{K}+1}} \\ &= g^{2^{\ell_\mathsf{K}+1} r \cdot 2\mathrm{dlog}_g \prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i}} = \left(R^2\right)^{2\mathrm{dlog}_g \prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i}}, \end{aligned}$$

so that consistency in the sense of (18) holds, and $\mathsf{Dec}'(sk, (R, S))$ outputs $\mathsf{BBS}_N(T) = \mathsf{BBS}_N((R^2)^{\frac{1}{2^{\ell_\mathsf{K}+1}}}) = \mathsf{BBS}_N(g^r) = K$.

Efficiency. With some trivial optimizations, KEM' uses roughly two exponentiations for encapsulation and one for decapsulation. However, KEM''s public key contains $2\ell_\mathsf{T} + 1$ group elements. This number can be roughly halved by using the following technique. Namely, observe that in KEM', the hash value $t = \mathsf{T}(R)$ is interpreted as a *bitwise* selector of ℓ_T out of $2\ell_\mathsf{T}$ group elements $X_{i,j}$. Instead, one can interpret t as an *integer* that specifies a subset of group elements X_i. Concretely, we can define a mapping f from $\{0,1\}^{\ell_\mathsf{T}}$ to subsets of $[\ell]$, where ℓ denotes the number of group elements. For our proof, we will only need that for any distinct $t, t^* \in \{0,1\}^{\ell_\mathsf{T}}$, we have $f(t) \not\subseteq f(t^*)$ and $f(t^*) \not\subseteq f(t)$. As an example, we can have the injective mapping f that associates to t the t-th subset of $[\ell]$ of size $\ell/2$. Using a suitable enumeration of these subsets, f can be implemented efficiently, and we will only need about $\ell \approx \ell_\mathsf{T} + \log \ell_\mathsf{T}$ group elements X_i to make f injective. More sophisticated examples of such mappings

f were suggested in the context of one-time signatures, see [37]. However, since our scheme KEM is far superior in public key size and (roughly) on par with KEM′ in terms of efficiency, we omit the details.

For concreteness let us again assume that one regular exponentiation with an exponent of length ℓ requires $1.5 \cdot \ell$ modular multiplications and that one squaring takes the same time as one multiplication. Let us further assume that $\ell_{\mathsf{N}} := \mathsf{bitlength}(N) = 1024$ and $\ell_{\mathsf{K}} = \ell_{\mathsf{T}} = 80$. Then encapsulation requires $3\ell_{\mathsf{N}} + \ell_{\mathsf{K}} + \ell_{\mathsf{T}} = 3232$ multiplications; decapsulation requires $1.5\ell_{\mathsf{N}} + \ell_{\mathsf{K}} = 1616$ multiplications. Hence in terms of efficiency KEM′ is slightly better than KEM, in particular for decapsulation. However, KEM′ has the drawback of large public key size.

Theorem 4 (IND-CCA security of KEM′). *Assume* T *is a target collision resistant hash function and the factoring assumption holds. Then* KEM′ *is secure in the* IND-CCA *secure in the sense of Definition 2.*

Given Theorem 2, it suffices to prove the following theorem.

Theorem 5 (IND-CCA adversary on KEM′ ⇒ BBS-distinguisher). *For every adversary* A *that* $(t_{\mathsf{KEM}'}, \epsilon_{\mathsf{KEM}'})$*-breaks* KEM′*'s* IND-CCA *property, there exists an algorithm* D *that* $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$ *breaks* BBS *and an adversary* B *that* $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$*-breaks* T*, such that*

$$t_{\mathsf{BBS}} \approx t_{\mathsf{T}} \approx t_{\mathsf{KEM}'} \qquad \epsilon_{\mathsf{BBS}} + \epsilon_{\mathsf{T}} + \frac{2\ell_{\mathsf{T}} + 3}{2^{k-1}} \geq \epsilon_{\mathsf{KEM}'}.$$

Proof. **Setting up the variables for simulation.** Assume an adversary A on KEM′'s IND-CCA security. We define a BBS-distinguisher D, which acts on input (N, z, V) as follows. D first uniformly picks an element $h \in \mathbb{QR}_N$, exponents $\beta_{i,j} \in [(N-1)/4]$ for $i \in [\ell_{\mathsf{T}}]$, $j \in \{0,1\}$, and sets

$$L = \mathrm{lcm}(1, \ldots, \ell_{\mathsf{T}}) \qquad g = h^{2L} \qquad R^* = z$$
$$t^* = (t_1^*, \ldots, t_{\ell_{\mathsf{T}}}^*) = \mathsf{T}(R^*) \qquad X_{i,t_i^*} = g^{\beta_{i,t_i^*} 2^{\ell_{\mathsf{K}}}} \qquad X_{i,1-t_i^*} = g^{\beta_{i,1-t_i^*} 2^{\ell_{\mathsf{K}}}} \cdot h.$$

Preparation of challenge ciphertext and key. To complete the definition of the challenge ciphertext $C^* = (R^*, S^*)$, D defines

$$S^* = \left| R^{* \sum_{i=1}^{\ell_{\mathsf{T}}} \beta_{i,t_i^*}} \right| \quad \left(= \left| R^{* \frac{\mathrm{dlog}_g \prod_{i=1}^{\ell_{\mathsf{T}}} X_{i,t_i^*}}{2^{\ell_{\mathsf{K}}}}} \right| \right) \tag{20}$$

such that (18) is met. Note that the (real) corresponding key K^* according to Dec′ is defined as

$$K^* = \mathsf{BBS}_N \left(\left(R^{*2} \right)^{\frac{1}{2^{\ell_{\mathsf{K}}+1}}} \right) = \mathsf{BBS}_N(z^{\frac{1}{2^{\ell_{\mathsf{K}}}}}) = \mathsf{BBS}_N(u). \tag{21}$$

D then invokes A with public key $pk = (N, g, (X_{i,j})_{i,j})$, challenge ciphertext $C^* = (R^*, S^*)$, and challenge key V. Note that V is either the real challenge key $\mathsf{BBS}_N(u)$, or it is a uniform string.

On the distribution of simulated public key and challenge ciphertext. The distribution of public key pk and challenge ciphertext C^* is almost identical in simulation and $\mathsf{IND\text{-}CCA}$ experiment. Concretely, the proof of the following lemma is very similar to the proof of Theorem 1, and we omit it.

Lemma 2. *There exists an event* $\mathsf{bad}_{\mathsf{key}}$ *such that, conditioned on* $\neg\mathsf{bad}_{\mathsf{key}}$*, public key* pk *and challenge ciphertext* C^* *are identically distributed in simulation and* $\mathsf{IND\text{-}CCA}$ *experiment. Furthermore,* $\mathsf{Pr}[\mathsf{bad}_{\mathsf{key}}] \leq \frac{2\ell_{\mathsf{T}}+3}{2^{k-1}}$*, both in the simulation and in the* $\mathsf{IND\text{-}CCA}$ *experiment.*

Thus, conditioned on $\neg\mathsf{bad}_{\mathsf{key}}$, D perfectly simulates A's input as in the real $\mathsf{IND\text{-}CCA}$ experiment if $V = \mathsf{BBS}_N(u) = \mathsf{BBS}_N(z^{\frac{1}{2^{\ell_{\mathsf{K}}}}})$, and as in the ideal $\mathsf{IND\text{-}CCA}$ experiment if V is random.

How to handle A's decryption queries. It remains to describe how D handles decryption queries of A as in the $\mathsf{IND\text{-}CCA}$ experiment. So say that A submits a ciphertext (R, S) for decryption. We may assume that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$. Let $t = (t_1, \ldots, t_{\ell_{\mathsf{T}}}) = \mathsf{T}(R)$. Write $d = |\{i : t_i \neq t_i^*\}| \in \{0, \ldots, \ell_{\mathsf{T}}\}$ for the Hamming distance of t and t^*, and abbreviate $\beta = \sum_{i=1}^{\ell_{\mathsf{T}}} \beta_{i,t_i}$. We call a ciphertext *consistent* iff the original decryption algorithm would not have rejected it. Hence, by (18), a ciphertext is consistent iff

$$S^2 = \left(R^2\right)^{\frac{\mathrm{dlog}_g \prod_{i=1}^{k} X_{i,t_i}}{2^{\ell_{\mathsf{K}}}}} \quad \left(= \left(R^2\right)^{\frac{\beta 2^{\ell_{\mathsf{K}}} + \mathrm{dlog}_g h^d}{2^{\ell_{\mathsf{K}}}}} = \left(R^2\right)^{\beta + \frac{d}{2^{\ell_{\mathsf{K}}+1} L}}\right) \tag{22}$$

Our goal will be to implement a consistency check using our setup of variables above, and to decrypt consistent ciphertexts as in the $\mathsf{IND\text{-}CCA}$ experiment.

How to detect inconsistent ciphertexts. We first describe how D detects inconsistent ciphertexts. Namely, (22) is equivalent to

$$\left(S^2 (R^2)^{-\beta}\right)^{2^{\ell_{\mathsf{K}}+1} L} = (R^2)^d \tag{23}$$

since exponentiating with $2^{\ell_{\mathsf{K}}+1} L$ is a bijection on $\mathbb{QR}_N$,[5] and the group elements on both sides of (22) are squares. On the other hand, D can easily check (23) and hence efficiently detect and reject inconsistent ciphertexts.

How to decrypt consistent ciphertexts. Now assume that C is consistent and $t \neq t^*$ (so that $1 \leq d \leq \ell_{\mathsf{T}}$). Then,

$$\mathsf{BBS}_N\left(\left(S^2 (R^2)^{-\beta}\right)^{\frac{L}{d}}\right) \overset{(23)}{=} \mathsf{BBS}_N\left(\left(R^2\right)^{\frac{1}{2^{\ell_{\mathsf{K}}+1}}}\right) = K. \tag{24}$$

[5] At this point we need that the modulus N consists of the product of two safe primes such that $\mathbb{QR}_N$ is a cyclic group whose order does not divide the integers $1, \ldots, \ell_{\mathsf{T}}$. Hence for KEM' it does not seem to be possible to avoid safe primes. (In contrast to KEM, cf. Section 5.)

Since $L/d \in \mathbb{N}$ by definition of $L = \mathrm{lcm}(1, \ldots, \ell_{\mathsf{T}})$, D can retrieve K efficiently using (24). Hence, D can decrypt all consistent ciphertexts satisfying $t \neq t^*$.

The case $t = t^*$. So let us turn to the case that $t = t^*$ and the ciphertext is consistent. Then, if $R = R^*$ holds, we have

$$S^2 \overset{(22)}{=} \left(R^2\right)^{\frac{\mathrm{dlog}_g \prod_{i=1}^{k} X_{i,t_i}}{2^{\ell_{\mathsf{K}}}}} \overset{(*)}{=} \left(R^{*2}\right)^{\frac{\mathrm{dlog}_g \prod_{i=1}^{k} X_{i,t_i^*}}{2^{\ell_{\mathsf{K}}}}} \overset{(20)}{=} S^{*2}, \tag{25}$$

where $(*)$ uses $R = R^*$ and $t = t^*$. Furthermore, $(R, S) \neq (R^*, S^*)$ implies $|S| = S \neq S^* = |S^*|$, so that $S \neq \pm S^*$ and $(S + S^*)(S - S^*) = S^2 - S^{*2} \overset{(25)}{=} 0 \bmod N$ yields a non-trivial factorization of N. Hence, D can efficiently factor N to solve its own input challenge (N, z, L) directly whenever $R = R^*$ and (R, S) is consistent.

On the other hand, if $\mathsf{T}(R) = t = t^* = \mathsf{T}(R^*)$ and $R \neq R^*$, then A has broken the target-collision resistance of T. Formally, let $\mathsf{bad}_{\mathsf{TCR}}$ denote the event that $t = t^*$ and $R \neq R^*$. If $\mathsf{bad}_{\mathsf{TCR}}$ occurs, D can safely give up, since

$$\mathsf{Pr}\left[\mathsf{bad}_{\mathsf{TCR}}\right] \leq \mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k) \tag{26}$$

for a suitable PPT adversary B on T that simulates D and A.

Summary of the decryption procedure. We summarize the decryption cases:

- inconsistent (R, S) (consistency check (23)⇔(22)⇔(18) not passed): reject,
- consistent (R, S) and $t \neq t^*$: decrypt using (24),
- consistent (R, S), $t = t^*$, and $R = R^*$: factor N (using $S \neq \pm S^*$ and $S^2 = S^{*2}$ by (25)),
- consistent (R, S), $t = t^*$, and $R \neq R^*$: give up simulation (A has found a T-collision).

Hence, also decryption is faithfully simulated unless $\mathsf{bad}_{\mathsf{TCR}}$ occurs.

Finishing the proof. We conclude that, unless $\mathsf{bad}_{\mathsf{TCR}}$ or $\mathsf{bad}_{\mathsf{key}}$ occurs, D perfectly simulates the real IND-CCA experiment upon input $V = \mathsf{BBS}_N(u)$, and the ideal IND-CCA experiment if V is random. If we let D output whatever the simulated experiment outputs, we obtain:

$$\begin{aligned}
&\left|\mathsf{Pr}\left[\mathsf{D}(N, z, \mathsf{BBS}_N(u)) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM}',\mathsf{A}}(k) = 1\right]\right| \leq \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{TCR}}\right] + \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{key}}\right] \\
&\left|\mathsf{Pr}\left[\mathsf{D}(N, z, U_{\{0,1\}^{\ell_{\mathsf{K}}}}) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM}',\mathsf{A}}(k) = 1\right]\right| \leq \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{TCR}}\right] + \mathsf{Pr}\left[\mathsf{bad}_{\mathsf{key}}\right].
\end{aligned} \tag{27}$$

Using Lemma 2 and (26), Theorem 5 follows from (27).

Realizing Hash-and-Sign Signatures under Standard Assumptions

Susan Hohenberger[1,⋆] and Brent Waters[2,⋆⋆]

[1] Johns Hopkins University
susan@cs.jhu.edu
[2] University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. Currently, there are relatively few instances of "hash-and-sign" signatures in the standard model. Moreover, most current instances rely on strong and less studied assumptions such as the Strong RSA and q-Strong Diffie-Hellman assumptions. In this paper, we present a new approach for realizing hash-and-sign signatures in the standard model. In our approach, a signer associates each signature with an index i that represents how many signatures that signer has issued up to that point. Then, to make use of this association, we create simple and efficient techniques that restrict an adversary which makes q signature requests to forge on an index no greater than $2^{\lceil \lg(q) \rceil} < 2q$. Finally, we develop methods for dealing with this restricted adversary. Our approach requires that a signer maintains a small amount of state — a counter of the number of signatures issued. We achieve two new realizations for hash-and-sign signatures respectively based on the RSA assumption and the Computational Diffie-Hellman assumption in bilinear groups.

1 Introduction

Digital signatures are a fundamental cryptographic primitive and a key building block in many larger systems. Typically, known constructions fall into one of two categories: the "tree"-based approach or the "hash-and-sign" approach. This latter paradigm generally yields more efficient constructions and shorter signatures, and represents what practitioners have come to expect. While many realizations of hash-and-sign signatures exist in the random oracle model (e.g., [16,33,28,4,29,7,19,18]), efficient schemes in the standard model are rare. Moreover, random oracle model schemes can be based on well-studied assumptions such as the discrete logarithm problem, Computational Diffie-Hellman and RSA.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

⋆ Supported by NSF CNS-0716142 and a Microsoft New Faculty Fellowship.

⋆⋆ Supported by NSF CNS-0524252, CNS-0716199, CNS-0749931; the US Army Research Office under the CyberTA Grant No. W911NF-06-1-0316; and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security. Portions of this work were done while this author was at SRI International.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 333–350, 2009.

However, known standard model schemes are often based on much stronger assumptions, such as Strong RSA [17,13], q-Strong Diffie-Hellman [6] and LRSW [10].[1] These assumptions allow the adversary a significant amount of flexibility in his ability to win, such as allowing him to choose a signing exponent (Strong RSA), compute a value relative to a chosen constant (q-Strong Diffie Hellman), or choose a random base value (LRSW). In each case, there are *many* possible correct answers to the adversary's challenge; in fact, exponentially many. This stands in high contrast to weaker, more studied assumptions, such as the discrete logarithm problem, Computational Diffie-Hellman and RSA, where there is only *one* correct answer for a given challenge. These assumptions which restrict the adversary to a single correct response seem inherently more reliable than their flexible counterparts. Thus, an important direction, in our opinion, is to push forward to practical, standard model schemes under standard assumptions.

One challenging aspect is that the security definition of signatures [21] inherently allows the adversary a great deal of flexibility; she wins if she outputs a forgery on *any* message not previously signed. Likewise, most existing "hash-and-sign" standard model schemes inherently enable the adversary a good deal of flexibility on which forgeries it can output and then the security is based on the hardness of a problem where there are *many* possible solutions. For example, consider the construction of Cramer and Shoup [17,13]; since the legitimate signer chooses a random prime from an exponentially large range, any proof must consider a forger that has the same flexibility in choosing the exponent and therefore the reduction is to the Strong RSA assumption (i.e, given (N, y), it is hard to produce *any* pair (e, x) such that $e > 1$ and $x^e \equiv y \mod N$).

In this work, we present a new avenue: design the scheme in such a way that *enforces* that any adversary output forgeries in some small set of categories that roughly grows with the number of signatures created so far. (E.g., A category could correspond to an RSA exponent used for verification in a prior signature.) Once the forger is restricted to a small set of categories, the simulator can guess where to program the challenge within this set. Alternatively, one can view our approach as restricting the adversary to a small forgery set and then employing *selectively-secure* techniques. The primary contribution of this work is the new method for restricting a forger.

Our Approach. Let us give the intuition behind our two constructions. At the core of our method, we associate with each signature an index i representing the number of signatures previously issued by the signer. The actual signer will only issue one signature per index. Roughly, the idea is to *efficiently* force the adversary to forge on a previously seen index value. To restrict the adversary, each signature is comprised of two logical components: a "core" signature on the message under index i and a second component that bounds the highest index number that the adversary might use. The most naive method would be to create a separate signature that signs the current index being used; however,

[1] One recent exception is the signature scheme due to Waters [34], which is provably secure under the CDH assumption in bilinear groups. However, this scheme suffers from a large public key size.

this would itself demand a secure signature scheme and lead to a circularity. To avoid this, the second component for a signature on index i will be a "signature" on $\lceil \lg(i) \rceil$. If we allow at most 2^λ signatures, then there are at most λ possible values for this second component and realizing it becomes simple. Moreover; the set of "allowable" indices is at most a factor of 2 times the number of signatures given out so far. It follows that any adversary must forge on a index set of roughly the same size as the number of signatures he has seen (or break the second component). Once we apply these techniques to force the adversary into this small index set, we are in a position to create a system based on weaker assumptions.

Let us illustrate this by describing a simplified version of our RSA-based construction. Let N be a Blum-Williams integer. The signer publishes a modulus N, a random value $v \in \mathbb{Z}_N^*$ and a hash function that enumerates a sequence of primes, i.e., let $H(i) = e_i$. To generate a signature using index i on message m, the signer creates a "core" signature on m using the signing exponent e_i^{-1} and then also gives out i and the $\lceil \lg(i) \rceil$-th square root of v. This ensures that an adversary that makes at most q queries *must* sign using one of the first $2^{\lceil \lg(q) \rceil} < 2q$ values of e_i (i.e., the output of $H(j)$ on $j = 1$ to $2^{\lceil \lg(q) \rceil}$); otherwise, the adversary can take square roots and thereby factor N. Now that the adversary is restricted to forging using a small set of e_i values, we can use a combination of previous techniques and new ideas to reduce from the standard RSA assumption.

Outline. We realize two new hash-and-sign signatures under the RSA assumption and the Computational Diffie-Hellman assumption in bilinear groups, in Sections 3 and 4 respectively. In Section 5, we discuss how to manage the signer's state in practice, including across multiple machines.

1.1 Related Work

The related work on designing secure signature schemes is vast and long standing. We provide only a brief summary.

Tree-Based. Many of the earliest provably-secure constructions used the design paradigm of a tree. Here a bound on the number of signatures to be issued is first established, and then the efficiency of the signatures (i.e., their size and the size of the public key) is in some way proportional to this bound. From general assumptions, a series of works including Bellare-Micali [2], Naor-Yung [27], and Rompel [32] established that signatures can be based on one-way functions. From general and concrete assumptions, another series of works sought more efficient solutions, such as those of Goldwasser-Micali-Rivest [21], Goldreich [20], Merkle [24], Dwork-Naor [15], Cramer-Damgård [11,12] and many more. While these works are fundamental to our understanding of provably secure signatures, the tree-based constructions are often passed over in practice due to the computation and memory requirements.

Hash-and-Sign. In the search for more efficient constructions, many schemes in the random oracle model were proposed, such as those of El Gamal [16],

Schnorr [33], Okamoto [28], Bellare-Rogaway [4], Pointcheval-Stern [29] and more recent short signatures by Boneh-Lynn-Shacham [7], signatures with tight-reductions to Diffie-Hellman by Goh-Jarecki-Katz-Wang [19], and the recent lattice-based signatures of Gentry-Peikert-Vaikuntanathan [18]. Unfortunately, the schemes are only known to be secure relative to the random oracle heuristic.

Drawing closer to our objective, some prior works have explored secure hash-and-sign signatures in the standard model. In 1999, Gennaro, Halevi and Rabin [17] introduced the first hash-and-sign construction secure in the standard model; its security depends on the Strong RSA assumption. Subsequent works also based on Strong RSA of Cramer-Shoup [13] and Camenisch-Lysyanskaya [9] improved the efficiency and added efficient protocols, respectively. (We will make use of a key reduction technique due to Cramer and Shoup later on.)

More recent works pushed for shorter signatures in the standard model, moving away from Strong RSA to more complex bilinear assumptions. Two such examples are the Boneh-Boyen [6] signatures based on q-Strong Diffie-Hellman (i.e., given a generator g of prime order p and the tuple $(g^x, g^{x^2}, \ldots, g^{x^q})$, it is hard to compute $(c, g^{1/(x+c)})$ for any $c \in \mathbb{Z}_p^*$) and the Camenisch-Lysyanskaya [10] signatures based on the interactive LRSW assumption.

While these standard model schemes are useful and efficient, their security depends on strong assumptions. In Strong RSA, q-Strong Diffie-Hellman and LRSW, there are *many* correct answers to any given challenge, allowing the adversary a significant amount of flexibility. This stands in sharp contrast to mild and restricted assumptions such as RSA and CDH.

To our knowledge, the only example prior to this work of a practical signature scheme secure in the standard model and under a mild complexity assumption is due to Waters [34], whose scheme is based on CDH in bilinear groups. The drawback of Waters' scheme compared to our scheme under the same assumption in Section 4 is that the public key requires $O(\lambda)$ group elements, where λ is the security parameter, whereas our public key requires $O(1)$ group elements. There exist variants of the Waters scheme (e.g., [26]) offering tradeoffs between the public key size and the concrete security level, but the asymptotic behavior remains the same.

Interpretting our Results. In this work, we limit ourselves to the standard practice of polynomial-time reductions. If we allowed super-polynomial reductions, it seems possible to interpret the Gennaro-Halevi-Rabin [17] and Cramer-Shoup [13] solutions as provably secure under RSA and the selectively-secure signatures of Boneh-Boyen [5] (as derived from their selectively-secure identity-based encryption scheme) as provably secure under CDH. Indeed, one alternative way of viewing our techniques is as a method for restricting a signature adversary so that selectively-secure schemes become (fully) adaptively-secure.

One can also view our results as a step toward realizing practical, standard model signatures under standard assumptions in a *stateless* manner. We remind the reader that many of the early tree-based signatures, such as the GMR signatures [21], also required the signer to keep a counter on the number of signatures issued. Subsequently, Goldreich [20] showed how to remove this dependence on state. We believe that a parallel outcome is possible here.

2 Background

2.1 Signature Schemes

Since we consider stateful signers, we slightly alter the signature algorithm specifications as follows:

KeyGen(1^λ) : the key generation algorithm outputs a keypair $(\mathrm{PK}, \mathrm{SK})$ and an initial state s.

Sign(SK, s, M) : the signing algorithm takes in a secret key SK, a state s, and a message M, and produces a signature σ.

Verify(PK, M, σ)**:** the verification algorithm takes in a public key PK, a message M, and a purported signature σ, and returns 1 if the signature is valid and 0 otherwise.

We use the standard security notion of *existential unforgeability with respect to chosen-message attacks* as formalized by Goldwasser, Micali and Rivest [21]. Here the adversary is given the public key and access to a signing oracle. The adversary is considered to be successful if she is able to produce a valid signature on any message not queried to the oracle.

2.2 Chameleon Hash Functions

A chameleon hash function $H(m, r)$ has the usual collision-resistant hash properties with the additional feature that, given some special trapdoor information, any target y and any message m', it is possible to efficiently find a value r' such that $H(m', r') = y$. Chameleon hash functions were formalized by Krawczyk and Rabin [23], who also presented a discrete-logarithm-based construction, derived from the chameleon commitments of Boyar et al. [8]. We employ this hash in Section 4 for our CDH-based signatures. In our RSA-based signatures in Section 3, we can employ any chameleon hash function. Secure constructions exist in the standard model under the discrete-logarithm assumption [23], the hardness of factoring [23], and the RSA assumption [1]. See Appendix A for more on RSA-based chameleon hashes.

2.3 RSA Assumption and Other Facts

We begin by recalling (one of the) standard versions of the RSA assumption [31].

Assumption 1 (RSA). *Let k be the security parameter. Let positive integer N be the product of two k-bit, distinct odd primes p, q. Let e be a randomly chosen positive integer less than and relatively prime to $\phi(N) = (p-1)(q-1)$. Given (N, e) and a random $y \in \mathbb{Z}_N^*$, it is hard to compute x such that $x^e \equiv y \mod N$.*

We remind the reader that in the **Strong RSA** assumption the adversary is given (N, y) and succeeds by producing any integer pair (e, x) such that $e > 1$ and $x^e \equiv y \mod N$. The standard RSA version is much more restrictive on the adversary.

We will make use of the following additional facts.

In our RSA-based scheme, we will require a primality test. Fortunately, for our purposes, it will be sufficient to use the efficient Miller-Rabin test [25,30].

Lemma 1 (Cramer-Shoup [13]). *Given $x, y \in \mathbb{Z}_n$ together with $a, b \in \mathbb{Z}$ such that $x^a = y^b$ and $\gcd(a, b) = 1$, there is an efficient algorithm for computing $z \in \mathbb{Z}_n$ such that $z^a = y$.*

Later on, we will choose our RSA moduli from the set of Blum-Williams integers, since we will require that each square has a unique square root which is itself a square. Formally, we will use:

Lemma 2 (Bellare-Miner [3]). *Suppose n is a Blum-Williams integer. Suppose $a, a_1, \ldots, a_t \in \mathbb{Z}_n^*$ and a is a square modulo n. Suppose $x, x_1, \ldots, x_t$ are integers such that $x_1, \ldots, x_t > x \geq 0$. Suppose*

$$a^{2^x} = \prod_{j=1}^{t} a_j^{2^{x_j}} \mod n \text{ , then } a = \prod_{j=1}^{t} a_j^{2^{x_j - x}} \mod n.$$

Theorem 2 (Prime Number Theorem). *Define $\pi(x)$ as the number of primes $\leq x$. For $x > 1$,*

$$\pi(x) > \frac{x}{\ln x}.$$

2.4 Bilinear Groups and the CDH Assumption

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p. A *bilinear map* is an efficient mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ which is both: (*bilinear*) for all $g \in \mathbb{G}$ and $a, b \leftarrow \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$; and (*non-degenerate*) if g generates $\mathbb{G}$, then $e(g, g) \neq 1$.

Assumption 3 (Computational Diffie-Hellman [14]). *Let g generate a group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$. For all p.p.t. adversaries $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr[a, b, \leftarrow \mathbb{Z}_p; z \leftarrow \mathcal{A}(g, g^a, g^b) : z = g^{ab}].$$

3 Our RSA Realization

In our later CDH construction, the signer's state of i will directly correspond to the ith signature. This will not be true here. In our simplified scheme in the introduction, we assumed the existence of a function that maps states to different prime exponents. Our first challenge is to realize this function in a way that allows us a means in the proof for embedding our RSA challenge exponent. Our realization of this function, denoted H below, will require that we skip over some states. Tantamount to our success will be the ability to maintain a correct distribution in our reduction.

To gain some intuition into the construction, let us begin by describing a *publicly-computable* hash function $H : \mathbb{Z} \to \{0,1\}^k$, which will be used to link exponents to states. Let $F : \mathbb{Z} \to \{0,1\}^k$ be a pseudorandom function family. Next, let c be a random value in $\{0,1\}^*$. For a random PRF key K, we define our corresponding hash function as $H_K(x) := c \oplus F_K(x)$.

While it is unusual to publicly release a PRF key, we do so because we only require some weaker properties from our hash function for which this construction will be sufficient. Specifically, we require that the hash function H_K: (1) outputs large primes with sufficient probability, and (2) on the first polynomial inputs to the hash, all prime outputs are distinct with high probability. We will later show that if the function H_K does not meet these requirements then F could not have been a PRF family.

Let us now turn to how the hash function H_K is used in the system. The signer keeps state as before and when signing with state s, if $H_K(s)$ is not prime, the signer will skip over it and increment its state until it reaches some s' such that $H_K(s')$ is prime. It will then sign using this prime and state s'. Thus, it will be important to guarantee that the signer not have to skip over too many indices when issuing signatures.

Now, we present our core construction and then remark on some different possible optimizations. The construction here already reduces the public key and signature size by one element in Z_N^* over the simplified RSA construction described in the introduction.

3.1 RSA Construction

Setup(1^λ). The setup algorithm chooses a Blum-Williams integer N, such that $2^\ell < \phi(N) < 2^{\ell+2}$, where ℓ is another security parameter derived from 1^λ. It then chooses two random quadratic residues $u, h \in \mathrm{QR}_N$.

Next, it establishes a hash function $H : \mathbb{Z} \to \{0,1\}^\ell$ by choosing a random key K for the PRF function $F : \mathbb{Z} \to \{0,1\}^\ell$, a random $c \in \{0,1\}^\ell$, and defining $H_K(x) = c \oplus F_K(x)$.

It publishes the parameters L of some Chameleon Hash scheme ChamHash : $\{0,1\}^{\ell'} \times \{0,1\}^{\ell''} \to \{0,1\}^{\frac{2\ell}{3}}$. (We note that such schemes in the standard model exist under the hardness of factoring [23] and RSA [1]; see Appendix A.)

Finally, the public key consists of

$$N, u, h, c, K, L.$$

Anyone can compute $H_K()$ using these parameters. The setup algorithm sets its state counter $s = 0$ and keeps the factorization of N as the secret key SK.

Sign(SK, $s, M \in \{0,1\}^{\ell'}$). The signer first increments its counter s by one as $s = s + 1$. The algorithm then chooses a random $r \in \{0,1\}^{\ell''}$ from the appropriate range dictated by the choice of ChamHash. It then computes $x = \mathrm{ChamHash}(M, r)$. Next, it checks if $H_K(s)$ is a prime. If not it increments $s = s+1$ and tries again until $e_s = H_K(s)$ is a prime. Then the signer computes:

$$B = (u^x h)^{(\frac{1}{2})^{\lceil \lg(s) \rceil}} \mod N$$

Note that we abuse notation here when taking square roots. When working modulo Blum-Williams integers, let $X^{\frac{1}{2}}$ represent the unique square root of X which is itself also a square. (See Lemma 2.) The signature is output as:

$$\sigma_1 = B^{\frac{1}{e_s}}, \quad r, \quad s.$$

Conceptually, s is an index, but we will skip over many s values where $H_K(s)$ is not a prime.

Verify$(\text{PK}, M, \sigma = (\sigma_1, r, i))$**.** The verification algorithm first makes sure that $i < 2^\lambda$. If it is greater, then it rejects. Second, the verifier checks that $H_K(i)$ is a prime. If not, it rejects.

Next, it squares σ_1 a total of $\lceil \lg(s) \rceil$ times yielding the value $Y = (\sigma_1)^{2^{\lceil \lg(s) \rceil}}$. Finally, it computes $x = \text{ChamHash}(M, r)$ and $e_i = H_K(i)$, and rejects unless it verifies that

$$Y^{e_i} \equiv (u^x h) \mod N.$$

Comments. The above scheme is geared to showcase the main ideas, however, one might consider different variants that allow for faster signature generation and faster verification. One area for improvement is in the way that prime exponents are generated and linked to states.

As a first variant, instead of having the signer skip over an index i if $H_K(i)$ is not prime (and thus, update and write to memory a state change), consider a scheme that allows the signer to search for a prime in a small range around the value $H_K(i)$. This option would require a more detailed analysis of the probability of a collision among the prime exponents used as well as a more complicated method for plugging in the RSA challenge.

As a second variant, consider a scheme that uses the first q primes starting with 3 to sign q messages. This variant would enable both faster generation (via a prime number seive to find the primes) and faster verification since we'd be using small prime exponents. Unfortunately, this appears to require a reduction from an assumption different than standard RSA; in particular, one might consider reducing from an assumption that the adversary can't take a root chosen randomly from the first q odd prime roots. For any polynomial q, this assumption is weaker than Strong RSA; however, we cannot reduce it to the standard RSA assumption.

A third avenue for optimization, focusing on the size of N and the chameleon hash parameters, is to note that our setting of $\{0,1\}^{\frac{2\ell}{3}}$ as the chameleon hash range is somewhat arbitrary. It can be set to any constant fraction of ℓ bits or any range R such that for a random prime $e \in \{0,1\}^\ell$ the probability that $e \notin R$ is non-negligible (we achieve $e \notin \{0,1\}^{\frac{2\ell}{3}}$ with high probability). In other words, there can be a tradeoff here between the size of the parameters and the concrete security. We also remind the reader that one can enlarge the domain of a chameleon hash by first applying a normal collision-resistant hash function [23].

A fourth avenue for optimization, this time focusing on the number of elements in the public key, is to find a method for directly embedding the chameleon hash function into the signature itself (as we do in our CDH scheme in Section 4).

3.2 Proof of Security

Theorem 4. *If the RSA assumption holds when N is a Blum-Williams integer, then the above construction is a secure signature scheme.*

Proof. Our reduction will only work on certain types of RSA challenges. We first describe this challenge set and then describe the reduction.

Our reduction will "throw out" all RSA challenges (N, e^*, y) where e^* is *not* an odd prime less than 2^ℓ. Fortunately, good challenges will occur with polynomial probability. By construction, $\phi(N) < 2^{\ell+2}$. We also know, by Theorem 2, that the number of primes $\leq 2^\ell$ is $\geq \frac{2^\ell}{\ell}$. Thus, a loose bound on the probability of e^* being a prime in the proper range is $(\frac{2^\ell}{\ell})/2^{\ell+2} = \frac{1}{4\ell}$.

Now, we describe the reduction. Suppose we have an adversary that makes at most $q(\lambda)$ queries where $q()$ is a polynomial. (We say q queries where it is clear from context.) We show that this adversary breaks RSA, on challenges (N, e^*, y) where N is a Blum-Williams integer and e^* is an odd prime $< 2^\ell$. An adversary can have two types of forgeries. Let x be the highest index on which the adversary obtains a signature from the signer (i.e., the index at which the qth prime appears).

Type I. The adversary forges for a message with index i greater than $2^{\lceil \lg(x) \rceil}$.
Type II. The adversary forges for a message with index i less than or equal to $2^{\lceil \lg(x) \rceil}$.

In Lemma 3, we show that a type I adversary can be used to break factoring with a loss of a λ factor in the reduction. In Lemma 4, we show that a type II adversary can be used to break RSA with a loss of a t factor in the reduction, where t is a polynomial-size "bound on $2^{\lceil \lg(x) \rceil}$" set to $4\ell[q + \lambda]$. The value t is established to avoid a circularity. In the proof of Lemma 4, the simulator would like to guess the index of the adversary's forgery in the range 1 to $2^{\lceil \lg(x) \rceil}$, so that it can set the function H_K accordingly. However, the simulator cannot compute x until it sets H_K. To avoid this circularity, we bound $t \geq 2^{\lceil \lg(x) \rceil}$. We want that for a random function R, there are at least q primes in the set of $\{R(i)\}_{i \in [1,x]}$ with high probability. Lemma 5 in Appendix B guarantees that this occurs for $x = 2\ell[q + \lambda]$ with probability $1 - e^{-[q+\lambda](\frac{1}{2})^2}$. Thus, we set $t = 2x = 4\ell[q + \lambda]$, which establishes that $t \geq 2^{\lceil \lg(x) \rceil}$. This concludes the proof.

Type I Adversary

Lemma 3. *If a type I adversary succeeds with probability ϵ, then it can be used to solve factoring with probability $\epsilon/(2\lambda) - negl(\lambda)$.*

Proof. We provide a reduction showing how to turn a type I adversary into an simulated adversary against factoring. Intuitively, in the simulation, the simulator takes a guess of $k^* = \lceil \lg(i) \rceil$, the logarithm of the index i on which the type I adversary will forge. There are at most λ values of k^* so the simulator has at least a $1/\lambda$ chance of guessing correctly.

We now describe the simulation. Given a factoring challenge $N = p_1 p_2$, where the goal is to produce either p_1 or p_2, proceed as follows.

Setup The simulator begins by guessing a value k^* in the range 1 to λ. Next, the simulator selects a random PRF key K and random $c \in \{0,1\}^\ell$, which defines the hash function $H_K()$. Next, for $i = 1$ to t, the simulator computes $e_i = H_K(i)$ and tests to see if it is prime. If e_i is prime, the simulator places i into a set E. If $|E| < q$, the simulator aborts. In Lemma 5, we show that due to our choice of t there will be at least q primes in E with high probability.

The simulator randomly chooses parameters L for a chameleon hash function, where $\{0,1\}^{\frac{2\ell}{3}}$ is the range of the hash. Finally, it chooses a random $u', h' \in \mathbb{Z}_N^*$ and sets

$$\hat{u} = (u')^{\prod_{j \in E} e_j} \quad \text{and} \quad \hat{h} = (h')^{\prod_{j \in E} e_j}$$
$$u = (\hat{u})^{2^{k^*}} \quad \text{and} \quad h = (\hat{h})^{2^{k^*}}.$$

Since u', h' are independently chosen, this will have the correct distribution.

The simulator outputs the public key as (N, u, h, c, K, L), sets the internal signing state $s = 0$ and keeps secret the chameleon hash trapdoor.

Sign When the adversary asks for a signature on message M, the simulator first updates its state value $s = s + 1$. Since the adversary is polynomial, we know that $s < 2^\lambda$. If $\lceil \lg(s) \rceil \geq k^*$, the simulator's guess was incorrect and it aborts. Otherwise, the simulator selects a random r, computes $x = \text{ChamHash}(M, r)$ and outputs the signature as:

$$\sigma_1 = \left((u'^x h') \prod_{j \in E}^{j \neq s} e_j \right)^{2^{(k^* - \lceil \lg(s) \rceil)}}, \quad r, \quad s.$$

Response Eventually, the type I adversary outputs a valid signature $\tilde{\sigma} = (\tilde{\sigma_1}, \tilde{r}, \tilde{i})$ on a message $\tilde{M}$. If $k^* \neq \lceil \lg(\tilde{i}) \rceil$, the simulator aborts. Otherwise, the simulator computes $x = \text{ChamHash}(\tilde{M}, \tilde{r})$. From the verification equation and simulation setup, we see that

$$(\tilde{\sigma_1}^{e^*})^{2^{k^*}} = u^x h = (\hat{u}^x \hat{h})^{2^{k^*}}.$$

Since N is a Blum-Williams integer, it follows that $(\tilde{\sigma_1^{e^*}})^2 = (\hat{u}^x \hat{h})^2$. Futhermore, the fact that h' was chosen randomly in $\mathbb{Z}_N^*$ and h' is raised to a product of odd primes e_i implies that with probability $\frac{1}{2}$ the value $\tilde{\sigma_1}^{e^*}$ is congruent to $\hat{u}^x \hat{h}$ modulo p_1, but not congruent modulo p_2 or vice versa. In this case, the simulator can factor N in the standard way, i.e., by computing a factor as $\mathbf{gcd}(\tilde{\sigma_1}^{e^*} - \hat{u}^x \hat{h}, N)$.

Type II Adversary

Lemma 4. *If a type II adversary succeeds with probability ϵ after making q signing queries, then it can be used to solve RSA where N is a Blum-Williams integer with probability $\epsilon/(4\ell[q + \lambda]) - negl(\lambda)$.*

Proof. We provide a reduction showing how to turn a type II adversary into an adversary against RSA. Our proof has two components. First, we'll describe a simulator and show that any adversary which is successful against the simulation can be used to break RSA. Second, we'll show that any adversary successful against the above signature scheme will also be successful against the simulation.

Intuitively, in the simulation, the simulator takes a guess of i^*, the index on which the type II adversary will forge, within a small range of 1 to t. We'll lose a factor of t here. The simulator will choose public parameters such that it is straightforward to sign for any index except i^*. If the simulator is asked to sign for index i^*, we program the Chameleon hash to allow this.

We now describe the simulation. Given an RSA challenge (N, e^*, y), where the goal is to produce a value $w \in \mathbb{Z}_N^*$ such that $w^{e^*} = y$, we proceed as follows. For notation, let $N = p_1 p_2$.

Setup. The simulator begins by guessing an index i^* in the range 1 to t. Next, the simulator selects a random PRF key K. It computes $c = F_K(i^*) \oplus e^*$, which defines the hash function $H_K()$. Recall that $e^* < 2^\ell$ since we "threw out" any other RSA challenge. Next, for $i = 1$ to t, the simulator computes $e_i = H_K(i)$ and tests to see if it is prime. If e_i is prime, the simulator places i into a set E. If $|E| < q$, the simulator aborts. In Lemma 5, we show that due to our choice of t there will be at least q primes in E with high probability.

The simulator randomly chooses parameters L for a chameleon hash function, where $\{0,1\}^{\frac{2\ell}{3}}$ is the range of the hash. The simulator then selects a random value $x^* \in \{0,1\}^{\frac{2\ell}{3}}$.

Finally, it chooses a random $d \in \mathbb{Z}_N^*$ and sets

$$\hat{u} = y^{\prod_{j \in E}^{j \neq i^*} e_j} \quad \text{and} \quad \hat{h} = \hat{u}^{-x^*} d^{\prod_{j \in E} e_j}, \text{ and then}$$
$$u = (\hat{u})^{2^\lambda} \quad \text{and} \quad h = (\hat{h})^{2^\lambda}.$$

Since y, d are independently chosen, this will have the correct distribution.

The simulator outputs the public key as (N, u, h, c, K, L), sets the internal signing state $s = 0$ and keeps secret the chameleon hash trapdoor.

Sign. When the adversary asks for a signature on message M, the simulator first updates its state value $s = s + 1$. Clearly, $s < 2^\lambda$. Now, there are two cases for computing the signature.

If $s = i^*$, then the simulator will employ the chameleon hash trapdoor to find a value r such that $\text{ChamHash}(M, r) = x^*$. The simulator then outputs the signature:

$$\sigma_1 = (d^{\prod_{j \in E}^{j \neq i^*} e_j})^{2^{(\lambda - \lceil \lg(i^*) \rceil)}}, \quad r, \quad i^*.$$

To verify correctness, notice that we can rewrite σ_1 as follows:

$$\begin{aligned}
\sigma_1 &= (\hat{u}^{x^*/e^*} \cdot \hat{u}^{-x^*/e^*} \cdot d^{\prod_{j \in E}^{j \neq i^*} e_j})^{2^{(\lambda - \lceil \lg(i^*) \rceil)}} \\
&= (\hat{u}^{x^*/e^*} \cdot \hat{h}^{1/e^*})^{2^{(\lambda - \lceil \lg(i^*) \rceil)}} \\
&= \left((\hat{u}^{x^*} \hat{h})^{1/e^*}\right)^{2^{(\lambda - \lceil \lg(i^*) \rceil)}} = ((u^{x^*} h)^{1/e*})^{(\frac{1}{2})^{\lceil \lg(i^*) \rceil}}
\end{aligned}$$

If $s \neq i^*$, then the simulator chooses a random r and computes the value $x = \text{ChamHash}(M, r)$. The simulator then outputs the signature:

$$\sigma_1 = \left((y^x \prod_{j \in E}^{j \neq s, j \neq i^*} e_j) \cdot (y^{-x^*} \prod_{j \in E}^{j \neq s, j \neq i^*} e_j) \cdot (d^{\prod_{j \in E}^{j \neq s} e_j}) \right)^{2^{(\lambda - \lceil \lg(s) \rceil)}}, \quad r, \quad s.$$

To verify correctness, notice that we can rewrite σ_1 as follows:

$$\sigma_1 = ((\hat{u}^x \hat{h})^{1/e_s})^{2^{(\lambda - \lceil \lg(s) \rceil)}} = ((u^x h)^{1/e_s})^{(\frac{1}{2})^{\lceil \lg(s) \rceil}}$$

Response. Eventually, the type II adversary outputs a valid signature $\tilde{\sigma} = (\tilde{\sigma_1}, \tilde{r}, \tilde{i})$ on a message $\tilde{M}$ such that $\tilde{i} \leq t$. If $\tilde{i} \neq i^*$, the simulator's guess was incorrect and it aborts.

Otherwise, the simulator computes $x = \text{ChamHash}(\tilde{M}, \tilde{r})$. If $x = x^*$, the simulator aborts. Otherwise, from the verification equation and simulation setup, we see that

$$(\tilde{\sigma_1}^{e^*})^{2^{\lceil \lg(\tilde{i}) \rceil}} = u^x h = (\hat{u}^x \hat{h}) 2^\lambda$$

We can see that $(\tilde{\sigma_1}^{e^*})^2 = (\hat{u}^x \hat{h})^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil + 1}}$ via Lemma 2 since $\lambda > \lceil \lg(\tilde{i}) \rceil$ and N is a Blum-Williams integer. Thus, we have two cases to consider regarding the underlying square roots.

- Case A: $\tilde{\sigma_1}^{e^*}$ or $-\tilde{\sigma_1}^{e^*}$ is equal to $(\hat{u}^x \hat{h})^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil}} \bmod N$.
- Case B: $\tilde{\sigma_1}^{e^*}$ is congruent to $(\hat{u}^x \hat{h})^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil}} \bmod p_1$, but not congruent mod p_2 or vice versa.

Case A: Suppose $v = \tilde{\sigma_1}^{e^*}$ is congruent to $(\hat{u}^x \hat{h})^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil}} \bmod N$. (The case for $-v$ is analogous.) Clearly, v is a square modulo N. Since $\lambda > \lceil \lg(\tilde{i}) \rceil$, we can apply Lemma 2 to obtain

$$\tilde{\sigma_1}^{e^*} = (\hat{u}^x \hat{h})^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil}}.$$

Let $v = \sigma_1 / (d^{\prod_{j \in E}^{j \neq i^*} e_j})^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil}}$. Then substituting into the above equation we have:

$$v^{e^*} = y^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil} (x - x^*) \prod_{j \in E}^{j \neq i^*} e_j}.$$

The simulator now runs the algorithm from Lemma 1 to obtain a value w such that $w^{e^*} = y$, and outputs w as the RSA challenge solution. We can apply Lemma 1 since (1) $w, y \in \mathbb{Z}_N$, (2) e^* and $2^{\lambda - \lceil \lg(\tilde{i}) \rceil}(x - x^*) \prod_{j \in E}^{j \neq i^*} e_j$ are in $\mathbb{Z}$, and (3) e^* is relatively prime to $2^{\lambda - \lceil \lg(\tilde{i}) \rceil}(x - x^*) \prod_{j \in E}^{j \neq i^*} e_j$ with high probability as shown in Lemma 6 of Appendix B.

Case B: The simulator computes $\mathbf{gcd}(\tilde{\sigma_1}^{e^*} - (\hat{u}^x \hat{h})^{2^{\lambda - \lceil \lg(\tilde{i}) \rceil}}, N)$ to obtain a factor of N.

This ends our description of the simulator. Due to space considerations, we leave to the full version [22] our argument that any successful type II adversary against our scheme will have success in the game presented by the simulator. To

do this, we first define a sequence of games, where the first game models the real world and the final game is exactly the view of the adversary when interacting with our simulator. We then show via a series of claims that if a type II adversary is successful against Game j, then it will also be successful against Game $j+1$.

4 Our CDH Realization

Our CDH construction is both simplier and more efficient than its RSA counterpart. This is partly due to the fact that here will not need to search for primes and can instead directly associate the ith state with the ith signature. Here we will also directly embed the chameleon hash function.

As before, each signature is associated with an index i and a category $k = \lceil \lg(i) \rceil$. We force the adversary to forge on a previously seen category, which restricts her to a polynomial-size set from which to choose her forgery index. Since this remaining set is polynomial in size, we can employ selectively-secure techniques to obtain an adaptively-secure scheme. Specifically, we make use of the selectively-secure signatures due to Boneh and Boyen [5] with a twist. Here our index is like the message in their scheme and our message impacts their "master key".

4.1 CDH Construction

Setup(1^λ)**.** The setup algorithm selects a bilinear group $\mathbb{G}$ of prime order $p > 2^\lambda$. It chooses a random exponent $a \in \mathbb{Z}_p$. It chooses random group elements $g, u, v, d, w, z, h \in \mathbb{G}$. The public key is output as:

$$g, g^a, u, v, d, w, z, h.$$

The setup algorithm sets its state counter $s = 0$ and keeps a as the secret key SK.

Sign$(\mathrm{SK}, s, M \in \mathbb{Z}_p)$**.** The message space is treated as $\mathbb{Z}_p$; to sign arbitrarily long messages one could first apply a collision-resistant hash function. The signer first increments its counter s by one as $s = s + 1$. If $s > 2^\lambda$, then abort. Otherwise, the algorithm chooses random $r, t \in \mathbb{Z}_p$ and then outputs the signature as:

$$\sigma_1 = (u^M v^r d)^a (w^{\lceil lg(s) \rceil} z^s h)^t, \quad \sigma_2 = g^t, \quad r, \quad s.$$

Conceptually, we can think of t as the randomness from the Boneh-Boyen selectively-secure signature [5] and r as the randomness for a Chameleon hash function $u^M v^r$.

Verify$(\mathrm{PK}, M, \sigma = (\sigma_1, \sigma_2, r, i))$**.** The verification algorithm first makes sure that $i < 2^\lambda$. If it is greater, then it rejects. Then it uses the bilinear map to verify the signature by checking that

$$e(\sigma_1, g) = e(u^M v^r d, g^a) e(\sigma_2, w^{\lceil \lg(i) \rceil} z^i h).$$

Theorem 5. *If the CDH assumption holds in $\mathbb{G}$, then the above construction is a secure signature scheme.*

Proof of this theorem appears in the full version [22].

Comments. First, the verification cost can be reduced to only two pairings by publishing or having each verifier do a one-time precomputation of the values $e(u, g^a), e(v, g^a)$ and $e(d, g^a)$. This is competitive with the most efficient bilinear schemes in the random oracle model, e.g., Boneh-Lynn-Shacham [7].

Second, we embedded a specific Chameleon hash function into our CDH scheme, because this appears to be the most efficient construction. We could, however, have set the signature as:

$$\sigma_1 = (u^x d)^a (w^{\lceil lg(s) \rceil} z^s h)^t, \quad \sigma_2 = g^t, \quad r, \quad s.$$

where $x = \text{ChamHash}(M, r)$ for *any* chameleon hash function mapping into $\mathbb{Z}_p$. However, the public parameters necessary for the chameleon hash would likely eclipse the gains of removing element v from the public key.

5 Handling State in Practice

One of the challenging issues when using our signature scheme in practice is that a signer must maintain state. Issues that may arise in practice include multiple (autonomous) machines sharing the same signing key and machine crashes, among other problems. Fortunately, in our scheme, since the state is a simple counter, most of these issues can be readily addressed.

Multiple Signers. Administrators often set up multiple machines with the same signing key (e.g., parallelizing SSL connections at a highly visited site). In most cases, it is impractical to assume that all of the machines can coordinate to maintain a shared state. However, in our system, there is a simple solution to deal with this problem. If n different machines are using the same signing key, then machine i can give its jth signature with index $n \cdot j + i$.

Handling Machine Crashes. On an actual implementation, it is important to commit the counter increment (to persistent memory) before giving out the signature. Otherwise, a crash might cause two signatures to be given out for the same counter and thereby compromise security. We observe that it is perfectly fine to skip over a small (i.e., polynomial) number of counters and recommend erring on the side of safety.

Using the Machine Clock as a State. Instead of having a signer maintain a state counter, one interesting alternative is to use the machine clock time as the signer's state. This can theoretically work in our system since the clock time monotonically increases at a polynomial rate. One concern, however, is that the signer should not issue more than one signature per clock period. Two potential circumstances where this could arise are either if somehow the clock time was

set backwards or the signing algorithm in two different invocations read the same clock value. This is especially relevant in the age of dual-core processors, where mechanisms are in place for managing access to shared memory, but there are not necessarily guarantees that two cores on the same chip would not read out the same clock value. Overall, using the clock as the state could be a risky design choice and a detailed analysis of the system implementation should be made before applying it.

6 Conclusion and Open Problems

We presented two practical hash-and-sign signatures based on RSA and CDH in bilinear groups in the standard model. We employed a new technique for restricting any adversary's ability to forge, which can be alternatively viewed as a mechanism for transforming selectively-secure techniques into adaptively-secure constructions. We view our stateful constructions here as a step toward realizing similar *stateless* signatures. Recall that early tree-based signatures (e.g., the GMR signatures [21]) had a stateful signer, until Goldreich [20] showed how to remove the state. Goldreich's techniques do not appear to apply directly here, but we are optimistic that similar progress can be made. While we focused on RSA and CDH based constructions, it would also be interesting to realize constructions under CDH in a non-bilinear group, lattices, or general assumptions.

Finally, we note that hash-and-sign signatures and their extensions with efficient protocols (e.g., [9,10]) have been useful for integrating into larger systems, such as anonymous credentials and e-cash. With this new building block and some additional work, one might be able to base these larger systems on more standard assumptions.

Acknowledgments

We thank Dan Boneh for valuable discussions, including suggesting the second variant of our RSA scheme described in Section 3.1. We also thank Giuseppe Ateniese, Matthew Green and the anonymous reviewers for helpful comments.

References

1. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
2. Bellare, M., Micali, S.: How to sign given any trapdoor function. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 200–215. Springer, Heidelberg (1990)
3. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS, pp. 62–73 (1993)
5. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319 (2004)
8. Boyar, J., Kurtz, S.A., Krentel, M.W.: A discrete logarithm implementation of perfect zero-knowledge blobs. Journal of Cryptology 2(2), 63–76 (1990)
9. Camenisch, J.L., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
10. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
11. Cramer, R., Damgård, I.B.: Secure signature schemes based on interactive protocols. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 297–310. Springer, Heidelberg (1995)
12. Cramer, R., Damgård, I.B.: New Generation of Secure and Practical RSA-Based Signatures. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 173–185. Springer, Heidelberg (1996)
13. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. ACM Trans. on Information and System Security 3(3), 161–185 (2000)
14. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
15. Dwork, C., Naor, M.: Universal one-way hash functions and their cryptographic applications. In: Symposium on the Theory of Computation, pp. 33–43 (1989)
16. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
17. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signatures without the random oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
18. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Symposium on the Theory of Computing, pp. 197–206 (2008)
19. Goh, E.-J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. J. of Cryptology 20(4), 493–514 (2007)
20. Goldreich, O.: Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 104–110. Springer, Heidelberg (1987)
21. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Computing 17(2) (1988)
22. Hohenberger, S., Waters, B.: Realizing hash-and-sign signatures under standard assumptions (2009), `http://eprint.iacr.org/2009/028`
23. Krawczyk, H., Rabin, T.: Chameleon signatures. In: Network and Distributed System Security Symposium (2000)
24. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
25. Miller, G.L.: Riemann's hypothesis and tests for primality. Journal of Computer and System Sciences 13, 300–317 (1976)
26. Naccache, D.: Secure and practical identity-based encryption, Cryptology ePrint Archive: Report 2005/369 (2005)

27. Naor, M., Yung, M.: An efficient existentially unforgeable signature scheme and its applications. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 234–246. Springer, Heidelberg (1994)
28. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
29. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
30. Rabin, M.O.: Probabilistic algorithm for testing primality. Journal of Number Theory 12, 128–138 (1980)
31. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Comm. of the ACM 21(2), 120–126 (1978)
32. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Symposium on the Theory of Computing, pp. 387–394. ACM, New York (1990)
33. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)
34. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A Chameleon Hash Function Based on RSA

While the chameleon hash function based on the hardness of factoring due to Krawczyk and Rabin [23] would be sufficient to rest our Section 3 construction entirely on the difficulty of RSA, we now present a more efficient option.

This construction is due to Ateniese and de Mederios [1]. Their work actually presents an identity-based chameleon hash, which we simplify, since we will not require the identity-based property. To obtain the identity-based feature, the authors employed a signature scheme secure in the random oracle model and proved the security of their scheme within this context. For completeness, we provide a proof of the basic hash function under RSA in the standard model in the full version [22], but the credit here should go to the prior work.

Let ℓ be a security parameter. Let N be an RSA modulus such that $2^\ell < \phi(N) < 2^{\ell+2}$. Choose a random, positive $e \in \{0,1\}^\ell$ which is relatively prime to $\phi(N)$ and a random $J \in \mathbb{Z}_N$. Set the public key as (N, e, J) and keep as the trapdoor the factorization of N as well as a value d such that $ed \equiv 1 \mod \phi(N)$.

The hash $H : \{0,1\}^{\frac{2\ell}{3}} \times \mathbb{Z}_N \to \mathbb{Z}_N$ is computed as

$$H(m, r) = J^m r^e \mod N.$$

The holder of the trapdoor can compute a collision for any message m' by solving the following equation for r':

$$J^m r^e = J^{m'} r'^e \text{ as } r' = r(J^d)^{m-m'} \mod N.$$

We note that the choice of $\{0,1\}^{\frac{2\ell}{3}}$ is somewhat arbitrary. It could be optimized to any constant fraction of ℓ bits or any range $\{0,1\}^{\ell'}$ such that the probability that $e \notin \{0,1\}^{\ell'}$ is non-negligible.

Theorem 6. *The above chameleon hash function is secure under the RSA assumption in the standard model.*

B Completing the RSA Proof: Lemmas 5 and 6

Lemma 5. *The probability that there are less than q prime numbers in a set of $2\ell[q+\lambda]$ independent, randomly chosen ℓ-bit numbers is $\leq e^{-[q+\lambda](\frac{1}{2})^2}$.*

Lemma 6. *Assuming that F is a PRF family, then in the proof of Lemma 4, the challenge exponent e^* and the simulator-produced value of $2^{\lambda-\lceil \lg(\tilde{i})\rceil}(x - x^*)\prod_{j\in E}^{j\neq i^*} e_j$ are relatively prime with high probability.*

Proof of these technical lemmas appear in the full version [22]. The proof of Lemma 5 uses Chernoff bounds (lower tail) to provide the given probability bound. In the proof of Lemma 6, our argument is based on certain statistical properties which are necessary, but not sufficient for a PRF. The fact that we only use these statistical properties and not the full power of the PRF explains why we are able to give out the seed in the construction.

A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks

Jan Camenisch[1], Nishanth Chandran[2], and Victor Shoup[3]

[1] IBM Research, work funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216483
[2] UCLA, work done while visiting IBM Research
[3] NYU, work done while visiting IBM Research, supported by NSF award number CNS-0716690

Abstract. Recently, at Crypto 2008, Boneh, Halevi, Hamburg, and Ostrovsky (BHHO) solved the long-standing open problem of "circular encryption," by presenting a public key encryption scheme and proving that it is semantically secure against key dependent chosen plaintext attack (KDM-CPA security) under standard assumptions (and without resorting to random oracles). However, they left as an open problem that of designing an encryption scheme that *simultaneously* provides security against both key dependent chosen plaintext *and* adaptive chosen ciphertext attack (KDM-CCA2 security). In this paper, we solve this problem. First, we show that by applying the Naor-Yung "double encryption" paradigm, one can combine any KDM-CPA secure scheme with any (ordinary) CCA2 secure scheme, along with an appropriate non-interactive zero-knowledge proof, to obtain a KDM-CCA2 secure scheme. Second, we give a concrete instantiation that makes use the above KDM-CPA secure scheme of BHHO, along with a generalization of the Cramer-Shoup CCA2 secure encryption scheme, and recently developed pairing-based NIZK proof systems. This instantiation increases the complexity of the BHHO scheme by just a small constant factor.

1 Introduction

Encryption is the oldest cryptographic primitive; indeed, cryptography used to be synonymous with encryption. Despite this, the right definition for the security of encryption schemes has still not been settled! The first formal definition of security for public key encryption was that of *semantic security* [17], which, loosely speaking, states that given an encryption of a message an adversary cannot learn any information about the message itself. As it turned out, this notion of security does not offer sufficient protection for most practical applications [6], as it does not take into account that an adversary could learn (partial information about) some plaintext when he has access to a decryption oracle. The subsequent stronger notion of security against chosen ciphertext attacks (CCA2 security [31]) takes this into consideration and gives an adversary access to a

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 351–368, 2009.

decryption oracle that will decrypt any ciphertext except a particular "challenge ciphertext". CCA2 security was considered the final answer with regard to the security of public key encryption schemes.

However, none of the above notions of security allow an adversary to obtain encryptions of secret keys or, more generally, functions of secret keys. Black, Rogaway, and Shrimpton formally defined such a notion, calling it *Key Dependent Message (KDM)* security [5]. A similar notion, called *circular security*, was earlier defined by Camenisch and Lysyanskaya [11] and used to prevent sharing of credentials. Both papers provided constructions in the random oracle model.

Without resorting to the use of random oracles, constructing a public key encryption scheme (practical or not) that is semantically secure against key dependent chosen plaintext attack (KDM-CPA) was a long-standing open problem. It was only recently that Boneh et al. [9] gave a construction of a KDM-CPA secure public key encryption scheme. They proved their scheme secure under the Decisional Diffie-Hellman (DDH) assumption. We will refer to their scheme as the BHHO scheme, which extends to obtain KDM-CPA security under the more general K-linear assumption [36,27] (which includes the DDH assumption for $K = 1$ and the DLIN assumption [8] for $K = 2$). However, Boneh et al. left as an open problem the construction of an encryption scheme that is simultaneously secure against key dependent chosen plaintext *and* chosen ciphertext attack (KDM-CCA2).

Our Contribution. In this paper, we solve this problem by constructing the first KDM-CCA2 secure public key encryption scheme that can be proved secure under standard assumptions, and without random oracles. In fact, we show that a variation of the Naor-Yung paradigm [30] allows one to combine any KDM-CPA secure encryption scheme and any regular CCA2 secure encryption scheme, together with a non-interactive zero knowledge (NIZK) proof [7], to obtain a KDM-CCA2 secure encryption scheme.

Moreover, we give a nearly practical instantiation of our general construction using the BBHO KDM-CPA scheme, a K-linear version [14,36,23] of the Cramer-Shoup [13] CCA2 scheme, and recently developed pairing-based NIZK proof systems [19,18,20]. In the BHHO scheme, a ciphertext is a couple of hundred group elements and our construction blows this up only be a small constant factor (two or three, depending on the cryptographic assumption one employs). For our construction, we need a pairing $e : \mathbb{G} \times \Gamma \to \mathbb{G}_{\mathrm{T}}$, and we prove security under the K-linear assumption in $\mathbb{G}$ and the L-linear assumption in Γ, for appropriate constants K and L (and we also need a collision-resistant hash function).

Motivational Example: Key-Wrap. The "key-wrap" problem motivates the need for KDM-CCA2 secure encryption in practice. The key-wrap mechanism is found, for instance, in cryptographic coprocessors such as IBM's Common Cryptographic Architecture [25] and RSA's Public Key Cryptographic Standards [33]. Cryptographic coprocessors are tamper-proof hardware tokens that process requests from applications to perform cryptographic tasks such as encryption, signing and so on. One can view these tokens as trusted hardware that stores keys of all users in the

system. When an application (or user) wishes to perform a cryptographic task, it authenticates itself to the token and the token processes the request. For the purpose of creating backup of data or to transport keys from one token to another, it is often desired to encrypt keys (also known as "key wrapping"). Naturally, when we encrypt private keys with other keys it might lead to a circularity. In other words, an adversary might get to see an encryption of a secret key sk_1 with public key pk_2 as well as an encryption of a secret key sk_2 with public key pk_1 (such circularity can in general be more complicated). Although one can circumvent this problem by maintaining a hierarchy of keys and/or by maintaining separate keys for the purpose of wrapping other keys, this is not always convenient or possible. In addition, since the hardware token performs decryption, an adversary may effectively have access to a decryption oracle.

Labeled Encryption. In many applications in which one uses a CCA2 secure encryption scheme, the notion of a *label* is very useful. Very briefly, a label consists of public data which is non-malleably attached to a ciphertext. In effect, it allows the encryptor to control the context in which a ciphertext is decrypted. This notion has been around for a long time, under various names, e.g., "indicator", "tag", "header", "associated data" [28,38,37,12,29,26,32]. While one can always implement the label mechanism by appending the label to the plaintext, this is often not the most practical way to achieve this.

Coming back to the key-wrap problem, a label may be used to describe the type of message being encrypted: if it encrypts a key, who the key belongs to, etc. When the hardware token decrypts a ciphertext labeled as a key, it can restrict the usage of the decrypted key; in particular, the token can ensure that such a key is only used within the token in appropriate ways (e.g., decryption, further key-wrap). Even if a token restricts the usage in this way, an adversary may attempt a chosen ciphertext attack by submitting an encryption of a key that actually belongs to Alice, and make it look like it belongs to Bob; moreover, perhaps the adversary is authorized to decrypt ciphertexts under Bob's key, which in effect allows him to decrypt ciphertexts encrypted under Alice's key. However, if labels are used as described above, CCA2 security will prevent such attacks from succeeding.

Because of their utility, we include labels in our definition of KDM-CCA2 security, and implement them in our construction. Moreover, we exploit the label mechanism for plain CCA2 encryption in our general construction to bind together the two ciphertexts and NIZK proof of the Naor-Yung paradigm. In particular, we shall see that the CCA2 encryption scheme we use directly support labels in a way that interacts very nicely with pairing-based NIZK techniques, leading to a conceptually simple and quite efficient concrete instantiation of our general construction.

Another use of labels is to enlarge the message space of a CCA2 encryption scheme: to encrypt a sequence of messages as a package, one can generate a key pair for a strongly secure one-time signature scheme, and then encrypt each message in the sequence using the verification key as a label, and then signing the whole sequence of ciphertexts. This application is convenient for us, because the BHHO scheme can only encrypt one bit of a secret key at a time.

Other related work. Backes, Pfitzmann and Scedrov [2] and Backes, Dürmuth and Unruh [1] considered KDM-CCA2 security of symmetric and asymmetric encryption schemes, respectively. They in fact define a notion of security stronger than we consider in our paper, by allowing the adversary to obtain some of the secret keys. They showed that RSA-OAEP ([4]) is secure in this sense in the random oracle model.

Halevi and Krawczyk [22] studied key-dependent message security (under the name key-dependent input (KDI) security) with respect to primitives such as pseudo-random functions (PRFs) and block ciphers. They showed that in the ideal-cipher model, KDI secure PRFs can be built if one restricts the functions of the key to be independent of the ideal-cipher. Further, they showed that this goal cannot be achieved in the standard model. On the positive side, they show that if one allows the PRF construction to depend on a fixed public value, but does not allow the function of the key to depend on this value, then KDI secure PRFs can be constructed in the standard model. Hofheinz and Unruh [24], constructed a symmetric key encryption scheme that achieves KDM-CPA security when an adversary can only make a bounded number of encryptions. Haitner and Holenstein [21] proved negative results for KDM-CPA security of encryption schemes when an adversary can query encryptions of specific functions of the secret key.

Outline of the paper. In §2, we give and discuss the definitions of KDM-CCA2, NIZK proofs, and strong one-time signatures, i.e., the ingredients of our generic construction, which is presented in §3.

In §4, we present concrete instantiations of our building blocks: We recall the BHHO KDM-CPA encryption scheme, the K-linear version of the Cramer-Shoup CCA2 encryption scheme, and Groth's strongly secure one-time signature scheme. As a service to the reader, we give a self-contained exposition of a simplified version of the NIZK proof system of Groth and Sahai [20] as it applies to linear equations over a group. This allows us to completely describe the instantiation of our construction and analyze its complexity.

In the full version of the paper [10], we discuss an alternative construction of KDM-CCA2 encryption that uses a CPA secure encryption scheme instead of a CCA2 secure encryption scheme but requires an NIZK proof system that provides (unbounded) simulation soundness [34,35]. In the full paper, we also show how to make the general NIZK proofs of [20] (unbounded) simulation sound, given a CCA2 secure encryption scheme that supports ciphertexts with labels, which again illustrates the power labels.

2 Preliminaries

2.1 Notation

When we say that an algorithm is *efficient*, we mean that the algorithm runs in probabilistic polynomial time in the security parameter. All our algorithms and functions take as input an implicit security parameter. When we say that a function is *negligible*, we mean that it is negligible in the implicit security parameter. Let $a\|b$ denote the concatenation of string a with string b.

2.2 Definition of KDM-CCA2 Security

Let **E** be a public key encryption system that supports ciphertexts with labels, which consists of three (probabilistic) efficient algorithms $\mathsf{EncKeyGen}$, E and D. $\mathsf{EncKeyGen}$ is a randomized key generation algorithm, that outputs a public key/secret key pair (pk, sk). The algorithm E takes as input a message m (from the message space $\mathcal{M}$), a public key pk and a label ℓ, and outputs a ciphertext $c := \mathsf{E}(pk, m, \ell)$. When we need to explicitly refer to the randomness in the encryption, we shall refer to an encryption of a message m with randomness r by $\mathsf{E}(pk, m, \ell; r)$. The decryption algorithm D takes as input a secret key sk, a ciphertext c, and a label ℓ, and either outputs a message m or `reject`. The (perfect) correctness condition is that (with probability one) $\mathsf{D}(sk, \mathsf{E}(pk, m, \ell), \ell) = m$ for all messages m, labels ℓ and (pk, sk) pairs output by $\mathsf{EncKeyGen}$.

When we use a public key encryption scheme **E** that does not support labels, we refer to the encryption and decryption algorithms of such a scheme by $\mathsf{E}(pk, m)$ and $\mathsf{D}(sk, c)$, respectively.

We extend the definition of key dependent message security from Black et al. [5] to the notion of security against chosen ciphertext attack ([30,31,15]). We will note that the standard definitions of public key encryption security are specific instances of this definition.

Let $\mathcal{S}$ denote the space of secret keys output by $\mathsf{EncKeyGen}$. As in [22] and [9], key-dependence is defined with respect to a fixed set of functions $\mathcal{C}$. Let $n > 0$ be an integer and let $\mathcal{C}$ be a finite set of functions $\mathcal{C} := \{f : \mathcal{S}^n \to \mathcal{M}\}$. KDM-security is defined with respect to $\mathcal{C}$ through the following two experiments between a challenger and an adversary $\mathcal{A}$. Let $\mathfrak{d} \in \mathcal{M}$ be a fixed (dummy) message in $\mathcal{M}$. Experiment b (where $b = 0, 1$) is defined as follows:

1. **Initialization phase:** In both experiments the challenger runs $\mathsf{EncKeyGen}()$ n times and obtains n key pairs $(pk_1, sk_1), (pk_2, sk_2), \cdots, (pk_n, sk_n)$. It sends the vector $(pk_1, pk_2, \cdots, pk_n)$ to $\mathcal{A}$.
2. **Query phase:** In both experiments, $\mathcal{A}$ may adaptively make the following two types of queries to the challenger.
   (a) **Encryption queries:** $\mathcal{A}$ can make a query of the form (i, f, ℓ), where $1 \leq i \leq n$, $f \in \mathcal{C}$ and ℓ is a label. The challenger responds by setting $m := f(sk_1, sk_2, \cdots, sk_n) \in \mathcal{M}$.
   In Experiment $b = 0$, it sets $c := \mathsf{E}(pk_i, m, \ell)$.
   In Experiment $b = 1$, it sets $c := \mathsf{E}(pk_i, \mathfrak{d}, \ell)$.
   In both experiments, the challenger sends c to $\mathcal{A}$.
   When the adversary $\mathcal{A}$ submits (i, f, ℓ) as an encryption query and the response of the challenger is c, we call (i, c, ℓ) a *target tuple*.
   (b) **Decryption queries:** In both experiments, $\mathcal{A}$ can make a query of the form (i, c, ℓ), where $1 \leq i \leq n, c$ is a string to be decrypted using secret key sk_i and ℓ is a label. The only restriction is that (i, c, ℓ) cannot be a (previous) target tuple. Note that c might not necessarily be a valid

ciphertext. That is, c might not be an output of $\mathsf{E}(pk_j, m, \ell)$ for some $1 \leq j \leq n, m \in \mathcal{M}$ and ℓ.
In both experiments, the challenger responds with $\mathsf{D}(sk_i, c, \ell)$.

3. **Final phase:** Finally, the adversary outputs a bit $b' \in \{0, 1\}$.

Definition 1 (KDM-CCA2). *A public key encryption scheme* $\mathbf{E}$ *is* KDM-CCA2 *secure with respect to* $\mathcal{C}$ *if* $\left|\Pr[W_0] - \Pr[W_1]\right|$ *is negligible for all efficient adversaries* $\mathcal{A}$*, where* W_b *is the event that* $\mathcal{A}$ *outputs* $b' = 1$ *in Experiment* b*.*

Note that the standard security definitions for public key encryption can be viewed as specific instances of the above general definition.

KDM-CPA: By restricting $\mathcal{A}$ from making any decryption queries, we get the definition of key-dependent message semantic security (KDM-CPA) as defined in [9].

CCA2: When we restrict the set of functions $\mathcal{C}$ from which $\mathcal{A}$ can draw f to the set of all constant functions on $\mathcal{S}^n \to \mathcal{M}$, we get the experiment for multiple message, multiple key CCA2 security, which is equivalent to the standard CCA2 security for a single message and single key (see [3]). If we further restrict $\mathcal{A}$ from making any decryption queries, we obtain the standard definition for semantic security (also see [3]).

Also note that, unlike regular CPA and CCA2 security, for both KDM-CPA and KDM-CCA2 security, one cannot reduce the attack game to a single encryption query and a single key pair.

We note that the definition of security by Backes et al. [1] is somewhat stronger in that it allows the adversary to obtain some secret keys. To benefit from this in practice, the users need to carefully keep track of which keys were compromised, and which keys are related to each other via key-wrap. In contrast, our definition pessimistically assumes that if one key is compromised then all potentially related keys should be considered compromised as well—which is probably more realistic.

2.3 Non-interactive Zero-Knowledge Proofs

Let R be a binary relation that is efficiently computable. For pairs of the form $(x, w) \in R$, x is referred to as the statement and w as the witness. Let $\mathcal{L} := \{x : (x, w) \in R \text{ for some } w\}$.

A non-interactive proof system for R consists of the following efficient algorithms: a common reference string (CRS) generation algorithm CRSGen, a prover P, and a verifier V. The CRSGen algorithm outputs the CRS denoted by $\mathfrak{C}$. P takes as input $\mathfrak{C}$, statement x, and witness w. It produces a proof $\mathfrak{p}$ if $(x, w) \in R$ and outputs `failure` otherwise. V takes as input $\mathfrak{C}$, x, and $\mathfrak{p}$. V outputs `accept` if it accepts the proof and `reject` otherwise.

Definition 2 (NIZK[7,16]). $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ *is a* non-interactive zero-knowledge (NIZK) proof system *for* R *if it has the following properties described below:*

Perfect Completeness: *For all* $\mathfrak{C}$ *output by* $\mathsf{CRSGen}()$*, for all* $(x, w) \in R$*, and for all* $\mathfrak{p} := \mathsf{P}(\mathfrak{C}, x, w)$*,* $\Pr[\mathsf{V}(\mathfrak{C}, x, \mathfrak{p})$ outputs `reject`$] = 0$*.*

Computational Soundness: *Consider the following game:*

1. $\mathsf{CRSGen}()$ *is run to obtain* $\mathfrak{C}$*, which is given to the adversary* $\mathcal{A}$*.*
2. $\mathcal{A}$ *responds with* $(x, \mathfrak{p})$*.*

$\mathcal{A}$ *wins the game if* $\mathsf{V}(\mathfrak{C}, x, \mathfrak{p}) =$ `accept` *and* $x \notin \mathcal{L}$*. Let* W *be the event that* A *wins the game. Then, for all efficient adversaries* $\mathcal{A}$*, we have* $\Pr[W]$ *is negligible.*

Computational Zero-knowledge: *Let* $\mathsf{S} := (\mathsf{S}_1, \mathsf{S}_2)$ *be a simulator running in polynomial time. Consider the following two experiments:*

Experiment 0**:** $\mathsf{CRSGen}()$ *is run and the output* $\mathfrak{C}$ *is given to* $\mathcal{A}$*.* $\mathcal{A}$ *is then given oracle access to* $\mathsf{P}(\mathfrak{C}, \cdot, \cdot)$*.*

Experiment 1**:** $\mathsf{S}_1()$ *generates* $\mathfrak{C}$ *and trapdoor* $\mathfrak{t}$*.* $\mathcal{A}$ *is given* $\mathfrak{C}$*, and is then given oracle access to* $\mathsf{S}'(\mathfrak{C}, \mathfrak{t}, \cdot, \cdot)$*, where* $\mathsf{S}'(\mathfrak{C}, \mathfrak{t}, x, w)$ *is defined to be* $\mathsf{S}_2(\mathfrak{C}, \mathfrak{t}, x)$ *if* $(x, w) \in R$ *and* `failure` *if* $(x, w) \notin R$*.*

Let W_i *be the event that* $\mathcal{A}$ *outputs* 1 *in Experiment* i*, for* $i = 0$ *or* 1*. Then, for all efficient adversaries* $\mathcal{A}$*, we have* $\left|\Pr[W_0] - \Pr[W_1]\right|$ *is negligible.*

Note that Blum et al. [7] give a weaker, "one time" definition of computational zero-knowledge, in which the adversary is allowed to see only one fake proof. However, because we cannot reduce KDM-security to an attack game with a single encryption query, this is insufficient for our purposes.

2.4 Strongly Secure One-Time Signatures

We also require a *strongly secure one-time signature scheme*. This is a signature scheme that satisfies the following security property: after obtaining the verification key and a signature $\mathfrak{s}$ on any message m of its choice, it is infeasible for an efficient adversary to generate any valid signature $\mathfrak{s}^*$ on any message m^* with $(\mathfrak{s}^*, m^*) \neq (\mathfrak{s}, m)$. See [10] for a more formal definition.

3 Generic Construction of a KDM-CCA2 Secure Scheme

In this section, we give a generic construction of KDM-CCA2 secure public key encryption scheme $\mathbf{E}$ with respect to a set of functions $\mathcal{C}$. We require the following building blocks: a public key encryption scheme $\mathbf{E}_{\mathrm{kdm}}$ that is KDM-CPA secure with respect to the set of functions $\mathcal{C}$; a regular CCA2 secure public key encryption scheme $\mathbf{E}_{\mathrm{cca}}$ that supports ciphertexts with labels; an NIZK proof system $\mathbf{P}$ for the language $\mathcal{L}_{\mathrm{eq}}$ consisting of the set of all pairs of ciphertexts that encrypt the same message using $\mathbf{E}_{\mathrm{kdm}}$ and $\mathbf{E}_{\mathrm{cca}}$; and a strongly secure one-time signature scheme $\mathbf{S}$.

At a high level, $\mathbf{E}$ is similar to the construction of [30,15]. To encrypt a message m, we generate a key-pair for the scheme $\mathbf{S}$, encrypt m using both $\mathbf{E}_{\text{kdm}}$ and $\mathbf{E}_{\text{cca}}$, where the label for $\mathbf{E}_{\text{cca}}$ will contain the verification key generated above (along with any input label). Using $\mathbf{P}$, we give a proof that both ciphertexts contain the same plaintext. We then sign the two ciphertexts as well as the proof using $\mathbf{S}$. The final ciphertext consists of the verification key, the two ciphertexts, the proof, and the signature.

3.1 Construction

We now formally describe the scheme $\mathbf{E} := (\mathsf{EncKeyGen}, \mathsf{E}, \mathsf{D})$ in detail. Let $\mathbf{E}_{\text{kdm}} := (\mathsf{EncKeyGen}_{\text{kdm}}, \mathsf{E}_{\text{kdm}}, \mathsf{D}_{\text{kdm}})$ (with key pair $(pk_{\text{kdm}}, sk_{\text{kdm}})$) and let $\mathbf{E}_{\text{cca}} := (\mathsf{EncKeyGen}_{\text{cca}}, \mathsf{E}_{\text{cca}}, \mathsf{D}_{\text{cca}})$ (with key pair $(pk_{\text{cca}}, sk_{\text{cca}})$). Let $\mathbf{S} := (\mathsf{SignKeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Let $\mathcal{L}_{\text{eq}}$ be the set of all triples (c_1, c_2, ℓ) such that

$$\exists\, m, r_1, r_2 : c_1 = \mathsf{E}_{\text{kdm}}(pk_{\text{kdm}}, m; r_1) \ \wedge\ c_2 = \mathsf{E}_{\text{cca}}(pk_{\text{cca}}, m, \ell; r_2).$$

Let $\mathbf{P} := (\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ be an NIZK proof system for $\mathcal{L}_{\text{eq}}$. Note that there maybe common system parameters that are used to define $\mathbf{E}_{\text{kdm}}, \mathbf{E}_{\text{cca}}$, and $\mathbf{P}$, and these are input to all associated algorithms. The encryption scheme $\mathbf{E}$ comprises of the following three algorithms.

$\mathsf{EncKeyGen}()$:

1. Run $\mathsf{EncKeyGen}_{\text{kdm}}()$ and $\mathsf{EncKeyGen}_{\text{cca}}()$ to obtain key pairs $(pk_{\text{kdm}}, sk_{\text{kdm}})$ and $(pk_{\text{cca}}, sk_{\text{cca}})$, respectively.
2. Run $\mathsf{CRSGen}()$ to generate the CRS $\mathfrak{C}$ of the NIZK proof system $\mathbf{P}$.

The public key is $pk := (pk_{\text{kdm}}, pk_{\text{cca}}, \mathfrak{C})$. The secret key is $sk := sk_{\text{kdm}}$.

$\mathsf{E}(pk, m, \ell)$:

1. Let $c_{\text{kdm}} := \mathsf{E}_{\text{kdm}}(pk_{\text{kdm}}, m; r_{\text{kdm}})$.
2. Run $\mathsf{SignKeyGen}()$ to generate key pair $(VK_{\text{ots}}, SK_{\text{ots}})$.
3. Let $c_{\text{cca}} := \mathsf{E}_{\text{cca}}(pk_{\text{cca}}, m, \ell \| VK_{\text{ots}}; r_{\text{cca}})$.
4. Let $\mathfrak{p}$ be the NIZK proof (using $\mathbf{P}$) for $(c_{\text{kdm}}, c_{\text{cca}}, \ell \| VK_{\text{ots}}) \in \mathcal{L}_{\text{eq}}$.
5. Let $c' := c_{\text{kdm}} \| c_{\text{cca}} \| \mathfrak{p}$ and let $\mathfrak{s} := \mathsf{Sign}_{SK_{\text{ots}}}(c')$.

Then $\mathsf{E}(pk, m, \ell) := c_{\text{kdm}} \| c_{\text{cca}} \| \mathfrak{p} \| VK_{\text{ots}} \| \mathfrak{s}$.

$\mathsf{D}(sk, c, \ell)$: Parse c as $c_{\text{kdm}} \| c_{\text{cca}} \| \mathfrak{p} \| VK_{\text{ots}} \| \mathfrak{s}$ (and output `reject` if this fails). Output `reject` if either $\mathsf{Verify}_{VK_{\text{ots}}}(c_{\text{kdm}} \| c_{\text{cca}} \| \mathfrak{p}, \mathfrak{s}) =$ `reject` or $\mathsf{V}(\mathfrak{C}, (c_{\text{kdm}}, c_{\text{cca}}, \ell \| VK_{\text{ots}}), \mathfrak{p}) =$ `reject`; otherwise, output $\mathsf{D}_{\text{kdm}}(sk, c_{\text{kdm}})$.

The (perfect) correctness of the public key encryption scheme $\mathbf{E}$ trivially follows from the (perfect) correctness of the scheme $\mathbf{E}_{\text{kdm}}$, (perfect) completeness of the proof system $\mathbf{P}$, and the (perfect) correctness of the signature scheme $\mathbf{S}$.

3.2 Proof of Security

Theorem 1. *Let* $\mathbf{E}_{\text{kdm}}$ *be a* KDM-CPA *secure scheme with respect to the set of functions* $\mathcal{C}$*. Let* $\mathbf{E}_{\text{cca}}$ *be a* CCA2 *secure scheme,* $\mathbf{S}$ *a strong one-time signature scheme, and* $\mathbf{P}$ *an* NIZK *proof system for* $\mathcal{L}_{\text{eq}}$*. Then* $\mathbf{E}$*, as constructed above, is a* KDM-CCA2 *secure scheme with respect to* $\mathcal{C}$*.*

PROOF. The proof is through a sequence of games. We first present a schematic description of the sequence of games used to prove that **E** is KDM-CCA2 secure. The underlined parts indicate what has changed in each game.

Game	Process encrypt query	Process decrypt query	justification
0	enc. (m, m); real $\mathfrak{p}$	dec. c_{kdm}	
1	enc. (m, m); real $\mathfrak{p}$	dec. $\underline{c_{\text{cca}}}$	soundness for **P**
2	enc. (m, m); $\underline{\text{fake}}$ $\mathfrak{p}$	dec. c_{cca}	ZK for **P**
3	enc. (m, m); fake $\mathfrak{p}$	dec. c_{cca}; $\underline{\text{special reject}}$	strong one-time sig. **S**
4	enc. $(m, \underline{\mathfrak{d}})$; fake $\mathfrak{p}$	dec. c_{cca}; special reject	CCA2 for $\mathbf{E}_{\text{cca}}$
5	enc. $(m, \mathfrak{d})$; fake $\mathfrak{p}$	dec. c_{cca}; $\underline{\text{no special reject}}$	strong one-time sig. **S**
6	enc. $(\underline{\mathfrak{d}}, \mathfrak{d})$; fake $\mathfrak{p}$	dec. c_{cca}	KDM-CPA for $\mathbf{E}_{\text{kdm}}$
7	enc. $(\mathfrak{d}, \mathfrak{d})$; $\underline{\text{real}}$ $\mathfrak{p}$	dec. c_{cca}	ZK for **P**
8	enc. $(\mathfrak{d}, \mathfrak{d})$; real $\mathfrak{p}$	dec. $\underline{c_{\text{kdm}}}$	soundness for **P**

The sequence of games involving the challenger Ch and adversary $\mathcal{A}$ are more formally described below. Let W_i be the event that $\mathcal{A}$ outputs 1 in Game i.

Game 0: This is the actual attack game, i.e., Experiment 0 in Definition 1. When responding to an encryption query, Ch encrypts the actual message m using both encryption schemes. The label for E_{cca} additionally contains VK_{ots} which Ch picks using $\mathsf{SignKeyGen}()$. Ch gives a real proof $\mathfrak{p}$ that both encryptions contain the same message. It produces the signature $\mathfrak{s}$ using SK_{ots}.

Game 1: This game is exactly like Game 0, except that when responding to a decryption query, Ch decrypts using secret key sk_{cca} instead of sk_{kdm}. It follows from the soundness of the proof system **P** that $\left|\Pr[W_1] - \Pr[W_0]\right|$ is negligible.

Game 2: This game is exactly like Game 1, except that when responding to an encryption query, Ch gives a simulated proof $\mathfrak{p}$ (using the trapdoor of the proof system) instead of a real proof. It follows from the zero-knowledge property of **P** that $\left|\Pr[W_2] - \Pr[W_1]\right|$ is negligible.

Game 3: This game is exactly like Game 2, except that when responding to a decryption query of the form (i, c, ℓ) from $\mathcal{A}$ such that $c = c_{\text{kdm}} \| c_{\text{cca}} \| \mathfrak{p} \| VK_{\text{ots}} \| \mathfrak{s}$, Ch first checks if there exists a target tuple of the form (i, c^*, ℓ), with $c^* = c^*_{\text{kdm}} \| c_{\text{cca}} \| \mathfrak{p}^* \| VK_{\text{ots}} \| \mathfrak{s}^*$ for some $c^*_{\text{kdm}}, \mathfrak{p}^*$ and $\mathfrak{s}^*$. If this is the case, then let c^* be the first such response by Ch. Now if $c^* \neq c$, then Ch rejects the encryption query. We call this the *special rejection rule*. It follows from the strong one-time security of the signature scheme **S** that Ch rejects via the special rejection rule only with negligible probability and hence $\left|\Pr[W_3] - \Pr[W_2]\right|$ is negligible.

In Game 3, Ch never decrypts a ciphertext that was contained in a target tuple using sk_{cca}. We can therefore make use of the CCA2 security of $\mathbf{E}_{\text{cca}}$.

Game 4: This game is exactly like Game 3, except that when responding to an encryption query, Ch encrypts the dummy message $\mathfrak{d}$ using E_{cca} but still encrypts the actual message m using E_{kdm}. It follows from the CCA2 security of $\mathbf{E}_{\text{cca}}$ that $\left|\Pr[W_4] - \Pr[W_3]\right|$ is negligible.

Game 5: This game is exactly like Game 4, except that when responding to a decryption query, Ch no longer follows the special rejection rule that was defined in Game 3. It follows from the strong one-time security of the signature scheme $\mathbf{S}$, that $\left|\Pr[W_5] - \Pr[W_4]\right|$ is negligible.

Game 6: This game is exactly like Game 5, except that when responding to an encryption query, Ch encrypts the dummy message $\mathfrak{d}$ using both encryption schemes. It follows from the KDM-CPA security of $\mathbf{E}_{\mathrm{kdm}}$ that $\left|\Pr[W_6] - \Pr[W_5]\right|$ is negligible.

Game 7: This game is exactly like Game 6, except that when responding to an encryption query, Ch gives a real proof $\mathfrak{p}$ that both encryptions contain the same message. It follows from the zero-knowledge property of $\mathbf{P}$ that $\left|\Pr[W_7] - \Pr[W_6]\right|$ is negligible.

Game 8: This game is exactly like Game 7, except that when responding to a decryption query, Ch decrypts using secret key sk_{kdm} instead of sk_{cca}. It follows from the soundness of the proof system $\mathbf{P}$ that $\left|\Pr[W_8] - \Pr[W_7]\right|$ is negligible. Game 8 is Experiment 1 in Definition 1.

Combining the different games, we get that $\left|\Pr[W_8] - \Pr[W_0]\right|$ is negligible, which proves Theorem 1. A more detailed proof can be found in [10]. □

Note that we used the computational soundness property of the proof system $\mathbf{P}$ only in Games 1 and 8 and in both these games, Ch only gave real proofs for true statements. Hence "plain" soundness of $\mathbf{P}$ is sufficient and we do not require the proof system to be simulation sound ([34]). In the definition of KDM-CCA2 security, one cannot reduce the attack game to a single encryption query and a single public key. Therefore, one-time zero-knowledge (see remark after Definition 2) would not be sufficient for our proof (one-time zero-knowledge does not imply multi-proof zero-knowledge). However, note that CCA2 security is sufficient, as the "single instance" definition implies the "multi-instance" definition (see remark after Definition 1).

4 Specific Number-Theoretic Instantiation of a KDM-CCA2 Secure Scheme

In this section, we give specific efficient instantiations of the building blocks used to construct the generic scheme presented in §3. We introduce notation and the number-theoretic assumptions in §4.1. In §4.2, we describe the KDM-CPA scheme of Boneh et al. [9], while in §4.3, we describe the K-linear version of the Cramer-Shoup CCA2 encryption scheme that we need. In §4.4 and §4.5, we describe the NIZK proof system used to prove equality of plaintexts. We use the efficient strongly one-time signature scheme of Groth [18] (which we describe in §4.6), to complete our instantiation of a KDM-CCA2 secure scheme. In §4.7, we discuss the size of the public key, system parameters, and ciphertext of our encryption scheme.

4.1 General Notation and Assumptions

Let $\mathbb{G}$ be a group of prime order q. We shall write $\mathbb{G}$ using multiplicative notation. One naturally views $\mathbb{G}$ as a vector space over $\mathbb{Z}_q$, where for $x \in \mathbb{Z}_q$ and $\mathbf{g} \in \mathbb{G}$, the "scalar product" of x and $\mathbf{g}$ is really the power $\mathbf{g}^x$. Because of this, we shall often employ concepts and terminology from linear algebra.

For vectors $\vec{\mathbf{g}} := (\mathbf{g}_1, \ldots, \mathbf{g}_R) \in \mathbb{G}^R$ and $\vec{x} := (x_1, \ldots, x_R) \in \mathbb{Z}_q^R$, define $\langle \vec{\mathbf{g}}, \vec{x} \rangle := \mathbf{g}_1^{x_1} \cdot \cdots \cdot \mathbf{g}_R^{x_R} \in \mathbb{G}$. When we write $\prod_{i=1}^{K} \vec{\mathbf{g}}_i \in \mathbb{G}^R$ for vectors $\vec{\mathbf{g}}_i \in \mathbb{G}^R$, we mean the component wise product of each of the R terms. Unless otherwise specified, there is no a priori relation between $\mathbf{g}, \vec{\mathbf{g}}, \mathbf{g}_i$ and $\vec{\mathbf{g}}_i$.

Definition 3 (K-linear assumption [36,27]). *Let $\mathbb{G}$ be a group of prime order q. For a constant $K \geq 1$, the K-linear assumption in $\mathbb{G}$ is defined through the following two experiments* (0 *and* 1) *between a challenger and an adversary $\mathcal{A}$ that outputs* 0 *or* 1.

Experiment 0**:** *The challenger picks $K+1$ random generators of $\mathbb{G}$: $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{K+1}$, picks random $x_1, \ldots, x_K \in \mathbb{Z}_q$ and sets $x_{K+1} = \sum_{i=1}^{K} x_i$. $\mathcal{A}$ is given $(\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{K+1}, \mathbf{g}_1^{x_1}, \mathbf{g}_2^{x_2}, \ldots, \mathbf{g}_{K+1}^{x_{K+1}})$ as input.*

Experiment 1**:** *The challenger picks $K+1$ random generators of $\mathbb{G}$: $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{K+1}$ and picks random $x_1, x_2, \ldots, x_{K+1} \in \mathbb{Z}_q$. $\mathcal{A}$ is given $(\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{K+1}, \mathbf{g}_1^{x_1}, \mathbf{g}_2^{x_2}, \ldots, \mathbf{g}_{K+1}^{x_{K+1}})$ as input.*

The K-linear assumption holds in $\mathbb{G}$ if for all efficient adversaries $\mathcal{A}$, $\left|\Pr[W_0] - \Pr[W_1]\right|$ is negligible, where W_i is the event that $\mathcal{A}$ outputs 1 *in Experiment i.*

Another way to understand the K-linear assumption is as follows. Let us define group vectors $\vec{\mathbf{g}}_1, \ldots, \vec{\mathbf{g}}_K \in \mathbb{G}^{K+1}$: $\vec{\mathbf{g}}_1 := (\mathbf{g}_1, 1, 1, \ldots, 1, \mathbf{g}_{K+1})$, $\vec{\mathbf{g}}_2 := (1, \mathbf{g}_2, 1, \ldots, 1, \mathbf{g}_{K+1}), \ldots,$ $\vec{\mathbf{g}}_K := (1, 1, \ldots, 1, \mathbf{g}_K, \mathbf{g}_{K+1})$. Let $\mathbb{T}$ denote the subspace of $\mathbb{G}^{K+1}$ generated by $\vec{\mathbf{g}}_1, \ldots, \vec{\mathbf{g}}_K$. The K-linear assumption says that it is hard to distinguish random elements of $\mathbb{T}$ from random elements of $\mathbb{G}^{K+1}$. Note that the standard Decisional Diffie-Hellman (DDH) assumption is the 1-linear assumption and the linear assumption (introduced in [8]) is the 2-linear assumption.

Pairings. Let $\mathbb{G}, \Gamma$ and $\mathbb{G}_{\mathrm{T}}$ be groups of prime order q. We shall use Roman letters to denote elements in $\mathbb{G}$ and Greek letters to denote elements in Γ. A *pairing* is a map $e : \mathbb{G} \times \Gamma \to \mathbb{G}_{\mathrm{T}}$ that satisfies the following properties: (1) e is *bilinear*, which means that for all $\mathbf{a} \in \mathbb{G}$ and $\alpha \in \Gamma$, the maps $e(\mathbf{a}, \cdot) : \Gamma \to \mathbb{G}_{\mathrm{T}}$ and $e(\cdot, \alpha) : \mathbb{G} \to \mathbb{G}_{\mathrm{T}}$ are linear maps; (2) e is *non-degenerate*, which means that its image is not $\{1\}$; and (3) e is *efficiently computable.*

4.2 KDM-CPA Secure Scheme Based on the K-Linear Assumption

In this section, we describe the public key encryption scheme of Boneh et al. [9] based on the K-linear assumption. Let $N := \lceil (K+2) \log_2 q \rceil$. $\mathbf{E}_{\mathrm{kdm}} = (\mathsf{EncKeyGen}_{\mathrm{kdm}}, \mathsf{E}_{\mathrm{kdm}}, \mathsf{D}_{\mathrm{kdm}})$ is as described below. The message space of this scheme is the group $\mathbb{G}$.

$\mathsf{EncKeyGen}_{\mathrm{kdm}}$:

1. Pick random $\vec{\mathbf{g}}_1, \ldots, \vec{\mathbf{g}}_K \in \mathbb{G}^N$.
2. Pick random $\vec{s} \in \{0,1\}^N$.
3. Define $\mathbf{h}_i := \langle \vec{\mathbf{g}}_i, \vec{s} \rangle \in \mathbb{G}$ for $i = 1, \ldots, K$.
4. Output the secret key $sk_{\mathrm{kdm}} := \vec{s}$ and the public key $pk_{\mathrm{kdm}} := (\vec{\mathbf{g}}_1, \ldots, \vec{\mathbf{g}}_K, \mathbf{h}_1, \ldots, \mathbf{h}_K)$.

$\mathsf{E}_{\mathrm{kdm}}(pk_{\mathrm{kdm}}, \mathbf{m})$:

1. Pick random $r_1, \ldots, r_K \in \mathbb{Z}_q$.
2. Output the ciphertext $(\vec{\mathbf{g}}, \mathbf{h}) := \left(\prod_{i=1}^K \vec{\mathbf{g}}_i^{r_i},\ \mathbf{m} \cdot \prod_{i=1}^K \mathbf{h}_i^{r_i}\right) \in \mathbb{G}^N \times \mathbb{G}$.

$\mathsf{D}_{\mathrm{kdm}}(sk_{\mathrm{kdm}}, (\vec{\mathbf{g}}, \mathbf{h}))$: Output $\mathbf{m} := \mathbf{h} / \langle \vec{\mathbf{g}}, \vec{s} \rangle$.

Note that the i^{th} bit s_i of the secret key $\vec{s}$ is encoded for the purpose of encryption as $\mathbf{g}^{s_i}$ for some random (but fixed) $\mathbf{g} \in \mathbb{G}$.

The key space (of encoded secret keys) is $\mathbb{G}^N$. Define a function $f_{\vec{t},\mathbf{b}} : \mathbb{G}^{nN} \to \mathbb{G}$ for fixed $\vec{t} \in \mathbb{Z}_q^{nN}$ and $\mathbf{b} \in \mathbb{G}$ to be the map $f_{\vec{t},\mathbf{b}}(\vec{\mathbf{u}}) := \langle \vec{\mathbf{u}}, \vec{t} \rangle \cdot \mathbf{b}$. Let $\mathcal{C}$ be the set of all functions $f_{\vec{t},\mathbf{b}}$ for all values of $\vec{t} \in \mathbb{Z}_q^{nN}$ and $\mathbf{b} \in \mathbb{G}$. $\mathbf{E}_{\mathrm{kdm}}$ is KDM-CPA secure with respect to the set of functions $\mathcal{C}$ [9].

Note that [9] explicitly describes the above scheme in the case $K = 1$, and only briefly mentions its generalization to $K > 1$ (the explicit description of which has been obtained from the authors of [9] via personal communication).

4.3 CCA2 Secure Scheme Based on the K-Linear Assumption

In this section, we describe a generalized version of the Cramer-Shoup encryption scheme based on the K-linear assumption. This generalization was described in [23] and [36]. However, given the K-linear decision problem, this scheme is essentially already implicit in [14] (based on Theorems 2 and 3, along with Example 1 in §7.4, of the full length version of that paper). This scheme is CCA2 secure and supports ciphertexts with labels. $\mathbf{E}_{\mathrm{cca}} = (\mathsf{EncKeyGen}_{\mathrm{cca}}, \mathsf{E}_{\mathrm{cca}}, \mathsf{D}_{\mathrm{cca}})$ is as described below. The message space of this scheme is the group $\mathbb{G}$, and the label space is $\{0,1\}^*$.

$\mathsf{EncKeyGen}_{\mathrm{cca}}$:

1. Pick random $\mathbf{f}_1, \ldots, \mathbf{f}_{K+1} \in \mathbb{G}$.
2. Pick random $\vec{x}, \vec{y}, \vec{z} \in \mathbb{Z}_q^{K+1}$.
3. Define the following elements of $\mathbb{G}^{K+1}$: $\vec{\mathbf{f}}_1 := (\mathbf{f}_1, 1, 1, \ldots, 1, \mathbf{f}_{K+1})$, $\vec{\mathbf{f}}_2 := (1, \mathbf{f}_2, 1, \ldots, 1, \mathbf{f}_{K+1})$, $\ldots$, $\vec{\mathbf{f}}_K := (1, 1, \ldots, 1, \mathbf{f}_K, \mathbf{f}_{K+1})$.
4. Define the following elements of $\mathbb{G}$: $\mathbf{c}_i := \langle \vec{\mathbf{f}}_i, \vec{x} \rangle$, $\mathbf{d}_i := \langle \vec{\mathbf{f}}_i, \vec{y} \rangle$, $\mathbf{e}_i := \langle \vec{\mathbf{f}}_i, \vec{z} \rangle$ $(i = 1, \ldots, K)$.
5. Output the secret key $sk_{\mathrm{cca}} := (\vec{x}, \vec{y}, \vec{z})$ and the public key $pk_{\mathrm{cca}} := \left(\{\mathbf{f}_j\}_{j=1}^{K+1}, \{\mathbf{c}_i\}_{i=1}^K, \{\mathbf{d}_i\}_{i=1}^K, \{\mathbf{e}_i\}_{i=1}^K\right)$.

$\mathsf{E}_{\mathrm{cca}}(pk_{\mathrm{cca}}, \mathbf{m}, \ell)$:

1. Pick random $w_1, \ldots, w_K \in \mathbb{Z}_q$.

2. Compute $(\vec{\mathbf{f}}, \mathbf{a}, \mathbf{b}) := \left(\prod_{i=1}^{K} \vec{\mathbf{f}}_i^{w_i},\ \mathbf{m} \cdot \prod_{i=1}^{K} \mathbf{c}_i^{w_i},\ \prod_{i=1}^{K} (\mathbf{d}_i \mathbf{e}_i^t)^{w_i}\right) \in \mathbb{G}^{K+1} \times \mathbb{G} \times \mathbb{G}$, where $t := H(\vec{\mathbf{f}}, \mathbf{a}, \ell) \in \mathbb{Z}_q$ and H is a collision resistant hash function. Output the ciphertext is $(\vec{\mathbf{f}}, \mathbf{a}, \mathbf{b})$.

$\mathsf{D}_{\mathrm{cca}}(sk_{\mathrm{cca}}, (\vec{\mathbf{f}}, \mathbf{a}, \mathbf{b}), \ell)$:

1. Verify that $\mathbf{b} = \langle \vec{\mathbf{f}}, \vec{y} + t\vec{z} \rangle$.
2. Output $\mathbf{m} := \vec{\mathbf{a}} / \langle \vec{\mathbf{f}}, \vec{x} \rangle$.

Note that the schemes in [14,36,23] do not explicitly support labels; however, the proof of security immediately generalizes to allow this, provided one assumes (as we do) that H is collision resistant.

4.4 NIZK Proofs for Satisfiable Systems of Linear Equations over Groups

In this section, we describe the NIZK proofs for proving that a system of linear equations over a group is satisfiable. These proofs are derived from Groth and Sahai [20]. The paper [20] deals with much more general systems of equations; for many applications, such as ours, we only need linear equations. For completeness, and concreteness, we describe how the methods of [20] apply to this setting. Our exposition is self contained, but brief.

Let $\mathbb{G}$ be a group of prime order q. A linear equation over $\mathbb{G}$ is an equation of the form $\mathbf{g}_0 = \prod_{j=1}^{W} \mathbf{g}_j^{\mathtt{X}_j}$, where $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_W \in \mathbb{G}$ are constants and $\mathtt{X}_1, \ldots, \mathtt{X}_W$ are variables. An *assignment* to the variables is a tuple $(x_1, \ldots, x_W) \in \mathbb{Z}_q^W$, and such an assignment *satisfies* the equation if $\mathbf{g}_0 = \prod_{j=1}^{W} \mathbf{g}_j^{x_j}$. A set S of linear equations over $\mathbb{G}$ is called *satisfiable* if there exists an assignment to the variables that simultaneously satisfies each equation in S.

Let $\mathcal{L}_{\mathrm{lsat}}$ be the language of all satisfiable sets of linear equations over $\mathbb{G}$. A witness for membership in $\mathcal{L}_{\mathrm{lsat}}$ is a satisfying assignment. Our goal is to construct an efficient NIZK proof system for $\mathcal{L}_{\mathrm{lsat}}$.

Our proof system for $\mathcal{L}_{\mathrm{lsat}}$ requires a pairing $e : \mathbb{G} \times \Gamma \to \mathbb{G}_{\mathrm{T}}$, where Γ and $\mathbb{G}_{\mathrm{T}}$ are also groups of order q. In addition, we need to make the L-linear assumption in Γ, for some constant L (typically, L is a small constant like 1 or 2, depending on the assumption we make).

- The CRS generator works as follows:
  1. Pick random $\gamma_1, \ldots, \gamma_{L+1} \in \Gamma$.
  2. Define the following elements of Γ^{L+1}: $\vec{\gamma}_1 := (\gamma_1, 1, \ldots, 1, \gamma_{L+1}), \vec{\gamma}_2 := (1, \gamma_2, \ldots, 1, \gamma_{L+1}), \ldots, \vec{\gamma}_L := (1, 1, \ldots, \gamma_L, \gamma_{L+1})$.
  3. Choose $\vec{\gamma} \in \Gamma^{L+1}$ at random.
  4. The common reference string is $(\gamma_1, \ldots, \gamma_{L+1},\ \vec{\gamma})$.
- Given a set S of equations, along with a satisfying assignment $(x_1, \ldots, x_W)$, the prover works as follows:
  1. Commit to $x_1, \ldots, x_W$ by setting $\vec{\delta}_j := \vec{\gamma}^{x_j} \prod_{k=1}^{L} \vec{\gamma}_k^{r_{jk}}$, for $j = 1, \ldots, W$, where the r_{jk}'s are randomly chosen elements of $\mathbb{Z}_q$.

2. The proof consists of the commitments $\vec{\delta}_1, \ldots, \vec{\delta}_W$, and, in addition, for each equation $\mathbf{g}_0 = \prod_{j=1}^{W} \mathbf{g}_j^{\mathtt{x}_j}$ in S, the proof contains L corresponding "proof elements" $\mathbf{p}_1, \ldots, \mathbf{p}_L \in \mathbb{G}$, which are computed as: $\mathbf{p}_k := \prod_{j=1}^{W} \mathbf{g}_j^{r_{jk}}$ $(k = 1, \ldots, L)$.

– To verify such a proof, the verifier takes the commitments $\vec{\delta}_1, \ldots, \vec{\delta}_W$, and, for each equation $\mathbf{g}_0 = \prod_{j=1}^{W} \mathbf{g}_j^{\mathtt{x}_j}$ in S, takes the corresponding proof elements $\mathbf{p}_1, \ldots, \mathbf{p}_L$, and checks that

$$\prod_{j=1}^{W} E(\mathbf{g}_j, \vec{\delta}_j) = E(\mathbf{g}_0, \vec{\gamma}) \prod_{k=1}^{L} E(\mathbf{p}_k, \vec{\gamma}_k). \tag{1}$$

Here, $E : \mathbb{G} \times \Gamma^{L+1} \to \mathbb{G}_{\mathrm{T}}^{L+1}$ is the biliear map that sends $(\mathbf{g}, (\alpha_1, \ldots, \alpha_{L+1}))$ to $(e(\mathbf{g}, \alpha_1), \ldots, e(\mathbf{g}, \alpha_{L+1}))$.

The CRS contains $2(L+1)$ elements of Γ, and a proof consists of $W(L+1)$ elements of Γ (for the commitments) and $|S|L$ elements of $\mathbb{G}$ (for the proof elements).

We now show that the above proof system has perfect completeness, (statistical) soundness, and computational zero-knowledge.

Perfect completeness. To argue perfect completeness, using bilinearity, one checks by a simple calculation that for any satisfying assignment $(x_1, \ldots, x_W)$, and for any choice of the r_{jk}'s, equation (1) will always be satisfied.

Soundness. A simple fact that will be useful in proving both the soundness and zero-knowledge property is the following, which the reader can easily verify using bilinearity:

Lemma 1. *If $\vec{\beta}_1, \ldots, \vec{\beta}_R \in \Gamma^{L+1}$ are linearly independent, then the map*

$$(\mathbf{h}_1, \ldots, \mathbf{h}_R) \mapsto E(\mathbf{h}_1, \vec{\beta}_1) \cdots E(\mathbf{h}_R, \vec{\beta}_R)$$

is an injective linear map from $\mathbb{G}^R$ into $\mathbb{G}_{\mathrm{T}}^{L+1}$.

To prove soundness, note that with overwhelming probability, the vectors $\vec{\gamma}, \vec{\gamma}_1, \ldots, \vec{\gamma}_L$ form a basis for Γ^{L+1}. Suppose a proof contains commitments $\vec{\delta}_1, \ldots, \vec{\delta}_W \in \Gamma^{L+1}$. Regardless of how these commitments were actually computed, each $\vec{\delta}_j$ can be expressed uniquely as $\vec{\delta}_j = \vec{\gamma}^{x_j} \prod_{k=1}^{L} \vec{\gamma}_k^{r_{jk}}$ for some $x_j, r_{j1}, \ldots, r_{jL} \in \mathbb{Z}_q$. Now consider any particular equation $\mathbf{g}_0^* = \prod_{j=1}^{W} \mathbf{g}_j^{\mathtt{x}_j}$, and corresponding proof elements $\mathbf{p}_1^*, \ldots, \mathbf{p}_L^*$. Define $\mathbf{g}_0 := \prod_{j=1}^{W} \mathbf{g}_j^{x_j}$ and $\mathbf{p}_k := \prod_{j=1}^{W} \mathbf{g}_j^{r_{jk}}$ for $k = 1, \ldots, L$, using the x_j's and r_{jk}'s determined as above by the commitments. On the one hand, by perfect completeness, we have $\prod_{j=1}^{W} E(\mathbf{g}_j, \vec{\delta}_j) = E(\mathbf{g}_0, \vec{\gamma}) \prod_{k=1}^{L} E(\mathbf{p}_k, \vec{\gamma}_k)$. On the other hand, if the verification equation (1) holds for the given equation and proof elements, then we also must have $\prod_{j=1}^{W} E(\mathbf{g}_j, \vec{\delta}_j) = E(\mathbf{g}_0^*, \vec{\gamma}) \prod_{k=1}^{L} E(\mathbf{p}_k^*, \vec{\gamma}_k)$. Thus, we have $E(\mathbf{g}_0, \vec{\gamma}) \prod_{k=1}^{L} E(\mathbf{p}_k, \vec{\gamma}_k) = E(\mathbf{g}_0^*, \vec{\gamma}) \prod_{k=1}^{L} E(\mathbf{p}_k^*, \vec{\gamma}_k)$. Applying Lemma 1 to the linearly independent vectors $\vec{\gamma}, \vec{\gamma}_1, \ldots, \vec{\gamma}_L$, we conclude that $\mathbf{g}_0 = \mathbf{g}_0^*$ (and in fact, $\mathbf{p}_k = \mathbf{p}_k^*$ for $k = 1, \ldots, L$). It follows that if the proof verifies, then the assignment $x_1, \ldots, x_W$ determined by the commitments simultaneously satisfies all the given equations.

Zero Knowledge. The simulator generates a "fake CRS" as follows: it generates $\vec{\gamma}_1, \ldots, \vec{\gamma}_L$ as usual, but it computes $\vec{\gamma}$ as $\prod_{k=1}^{L} \vec{\gamma}_j^{s_j}$ for random $s_1, \ldots, s_L \in \mathbb{Z}_q$. The trapdoor for the fake CRS is $(s_1, \ldots, s_L)$.

In a fake CRS, $\vec{\gamma}_1, \ldots, \vec{\gamma}_L$ are linearly independent (with overwhelming probability), while $\vec{\gamma}$ is a random element of the subspace V generated by $\vec{\gamma}_1, \ldots, \vec{\gamma}_L$.

To simulate a proof for a satisfiable set S of linear equations, the simulator starts by setting $\vec{\delta}_j := \prod_{k=1}^{L} \vec{\gamma}_k^{r_{jk}}$ for random $r_{jk} \in \mathbb{Z}_q$ for $j = 1, \ldots, W$ and $k = 1, \ldots, L$. For each equation $\mathbf{g}_0 = \prod_{j=1}^{W} \mathbf{g}_j^{\mathbf{x}_j}$ in S, the simulator generates proof elements $\mathbf{p}_1, \ldots, \mathbf{p}_L$ as follows: $\mathbf{p}_k := \mathbf{g}_0^{-s_k} \prod_{j=1}^{W} \mathbf{g}_j^{r_{jk}} \quad (k = 1, \ldots, L)$. The reader may easily verify, using the bilinearity property, that the verification equation (1) is satisfied.

We now argue that fake proofs are computationally indistinguishable from real proofs. To this end, let us introduce a hybrid prover, which works exactly like a real prover, except that it uses a fake CRS. Such hybrid proofs are computationally indistinguishable from real proofs, under the L-linear assumption for Γ. Moreover, hybrid proofs are statistically indistinguishable from fake proofs. To see this, observe that with overwhelming probability, $\vec{\gamma}_1, \ldots, \vec{\gamma}_L$ are linearly independent. Assuming this is true, in both the hybrid and fake proofs, the distribution of the commitments are the same (uniformly and independently distributed over the subspace V). Additionally, in both types of proofs, the proof elements $\mathbf{p}_1, \ldots, \mathbf{p}_L$ for a given equation are uniquely determined in the same way by the equation, the commitments, and the CRS; indeed, both types of provers generate proof elements that satisfy the verification equation (1); moreover, applying Lemma 1 to the vectors $\vec{\gamma}_1, \ldots, \vec{\gamma}_L$, we see that for a fixed equation, commitments, and CRS, there exist unique $\mathbf{p}_1, \ldots, \mathbf{p}_L$ that satisfy (1).

4.5 NIZK Proof for Proving Equality of Plaintext

Given a ciphertext of $\mathbf{E}_{\text{kdm}}$ (from §4.2) of the form $(\vec{\mathbf{g}}, \mathbf{h}) \in \mathbb{G}^N \times \mathbb{G}$ and a ciphertext of $\mathbf{E}_{\text{cca}}$ (from §4.3) of the form $(\vec{\mathbf{f}}, \mathbf{a}, \mathbf{b}) \in \mathbb{G}^{K+1} \times \mathbb{G} \times \mathbb{G}$ with respect to a label $\ell \in \{0,1\}^*$, we want to prove that they are valid encryptions of the same message. This is done by proving that there exist $r_1, \ldots, r_K,\ w_1, \ldots, w_K \in \mathbb{Z}_q$ such that $\vec{\mathbf{g}} = \prod_{i=1}^{K} \vec{\mathbf{g}}_i^{r_i}$, $\vec{\mathbf{f}} = \prod_{i=1}^{K} \vec{\mathbf{f}}_i^{w_i}$, $\mathbf{b} = \prod_{i=1}^{K} (\mathbf{d}_i \mathbf{e}_i^t)^{w_i}$, and $\mathbf{h}/\mathbf{a} = \prod_{i=1}^{K} \mathbf{h}_i^{r_i} / \prod_{i=1}^{K} \mathbf{c}_i^{w_i}$, where $t := H(\vec{\mathbf{f}}, \mathbf{a}, \ell)$.

This translates into $N+(K+1)+1+1 = N+K+3$ equations in $2K$ variables. Using the proof system above, this means we need $(2K)(L+1)$ elements of Γ for commitments, and $(N+K+3)L$ elements of $\mathbb{G}$ for the proofs.

4.6 Strongly Secure One-Time Signature Scheme

Here is the strongly secure one-time signature scheme $\mathbf{S}$ from Groth [18]. It makes use of a group $\mathbb{G}$ of prime order q with generator $\mathbf{g}$, and a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$. The scheme is secure assuming the hardness of computing discrete logs in $\mathbb{G}$ (which follows from the K-linear assumption) and assuming H is collision resistant.

SignKeyGen: Pick random $x, y \in \mathbb{Z}_q^*$ and $r, s \in \mathbb{Z}_q$, and set $\mathbf{f} := \mathbf{g}^x$, $\mathbf{h} := \mathbf{g}^y$, and $\mathbf{c} := \mathbf{f}^r\mathbf{h}^s$. The verification key is $VK = (\mathbf{f}, \mathbf{h}, \mathbf{c})$ and the secret key $SK = (x, y, r, s)$.

$\mathsf{Sign}_{SK}(m)$: To sign a message $m \in \{0,1\}^*$, pick t at random from $\mathbb{Z}_q$. The signature is $\mathfrak{s} = (t, (x(r-t) + ys - H(m))/y)$.

$\mathsf{Verify}_{VK}(m, \mathfrak{s})$: To verify the signature $\mathfrak{s} = (t, w)$, check that $\mathbf{c} = \mathbf{g}^{H(m)}\mathbf{f}^t\mathbf{h}^w$.

4.7 Size of Public Key, System Parameters and Ciphertext

Using the K-linear assumption for $\mathbb{G}$ and the L-linear assumption for Γ, the size of the public key, system parameters and ciphertext are as follows, where $N := \lceil (K+2)\log_2 q \rceil$.

The system parameters consists of the CRS which comprises $2(L+1)$ elements of Γ, the descriptions of $\mathbb{G}, \Gamma, \mathbb{G}_{\mathrm{T}}, e$ and the collision resistant hash function H for $\mathbf{E}_{\mathrm{cca}}$ and $\mathbf{S}$.

The public key of $\mathbf{E}$ consists of $(N+1)K$ elements of $\mathbb{G}$ for the public key pk_{kdm} and $4K+1$ elements of $\mathbb{G}$ for the public key pk_{cca}, for a total of $(N+5)K+1$ elements of $\mathbb{G}$.

The two ciphertexts (c_{kdm} and c_{cca}) require $(N+1)$ and $(K+3)$ elements of $\mathbb{G}$, respectively, giving a total of $N+K+4$ elements of $\mathbb{G}$. To prove equality of plaintexts, we require $(2K)(L+1)$ elements of Γ for commitments, and $(N+K+3)L$ elements of $\mathbb{G}$ for the proofs. Finally, to sign the resulting ciphertexts and proofs using the one-time signature scheme $\mathbf{S}$, we require 3 elements of $\mathbb{G}$ for the verification key VK of $\mathbf{S}$ and 2 elements of $\mathbb{Z}_q$ for the signature.

Note that we can make the public key shorter, by making pk_{cca} as part of the system parameters; indeed, since the secret key sk_{cca} is not needed (other than in the proof of security), one can simply generate all of the group elements appearing in pk_{cca} at random (yielding a distribution that is statistically close to the real distribution on public keys).

We emphasize that, typically, one would set $K = 1, 2$ and $L = 1, 2$, depending on the groups $\mathbb{G}$ and Γ. For example, at one extreme, if $\mathbb{G} = \Gamma$, then one could set $K = L = 2$; at the other extreme, if $\mathbb{G} \neq \Gamma$, and there is no (known) efficiently computable homomorphism from $\mathbb{G}$ to Γ or *vice versa*, then one could set $K = L = 1$.

References

1. Backes, M., Dürmuth, M., Unruh, D.: OAEP is secure under key-dependent messages. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 506–523. Springer, Heidelberg (2008)
2. Backes, M., Pfitzmann, B., Scedrov, A.: Key-dependent message security under active attacks - BRSIM/UC-soundness of symbolic encryption with key cycles. In: CSF, pp. 112–124 (2007)
3. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)

4. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
5. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Selected Areas in Cryptography, pp. 62–75 (2002)
6. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
7. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC 1988, pp. 103–112 (1988)
8. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
9. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision diffie-hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008)
10. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. Cryptology ePrint Archive, Report 2008/375 (2008), http://eprint.iacr.org/
11. Camenisch, J.L., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
12. Camenisch, J.L., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
13. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
14. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, p. 45. Springer, Heidelberg (2002), http://eprint.iacr.org/2001/085
15. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: STOC 1991, pp. 542–552 (1991)
16. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: FOCS 1990, pp. 308–317 (1990)
17. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: STOC 1982, pp. 365–377 (1982)
18. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
19. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
20. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
21. Haitner, I., Holenstein, T.: On the (im)possibility of key dependent encryption. In: TCC 2009 (2009)
22. Halevi, S., Krawczyk, H.: Security under key-dependent inputs. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 466–475. ACM, New York (2007)

23. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
24. Hofheinz, D., Unruh, D.: Towards key-dependent message security in the standard model. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 108–126. Springer, Heidelberg (2008)
25. IBM. IBM CCA Basic Services Reference and Guide for the IBM 4758 PCI and IBM 4764 PCI-X Cryptographic Coprocessors: Releases 2.53, 2.54, 3.20, 3.23, 3.24, 3.25, 3.27, and 3.30 (2008)
26. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
27. Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed diffiehellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
28. Lim, C.H., Lee, P.J.: Another method for attaining security against adaptively chosen ciphertext attacks. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 420–434. Springer, Heidelberg (1994)
29. MacKenzie, P.D., Reiter, M.K., Yang, K.: Alternatives to non-malleability: Definitions, constructions, and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 171–190. Springer, Heidelberg (2004)
30. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC 1990, pp. 427–437 (1990)
31. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
32. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)
33. RSA Laboratories. PKCS #11 v2.20: Cryptographic Token Interface Standard (2004)
34. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS 1999, pp. 543–553 (1999)
35. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
36. Shacham, H.: A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007), `http://eprint.iacr.org/`
37. Shoup, V.: A proposal for an ISO standard for public key encryption, version 2.1 (2001), `http://shoup.net/papers/`
38. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)

Cryptography without (Hardly Any) Secrets ?

Shafi Goldwasser

MIT and Weizmann Institute

Abstract. The absolute privacy of the secret-keys associated with cryptographic algorithms has been the corner-stone of modern cryptography. Still, in practice, keys do get compromised at times for a variety or reasons. A particularly disturbing loss of secrecy is as a result of side channel attacks. These attacks exploit the fact that every cryptographic algorithm is ultimately implemented on a physical device and such implementations enable 'observations' which can be made and measured on secret data and secret keys. Indeed, side channel observations can lead to information leakage about secret keys, which in turn can and have lead to complete breaks of systems which have been proved mathematically secure, without violating any of the underlying mathematical principles or assumptions. Traditionally, such attacks have been followed by ad-hoc 'fixes' which make particular implementation invulnerable to particular attacks, only to potentially be broken anew by new examples of side-channel attacks.

In recent years, starting with the work on *physically observable cryptography* by [MR04] Micali and Reyzin, a new goal has been set to build a general theory of physical security against a large class of families of side channel attacks which one may call *computational* side-channel attacks. These include *any* side channel attack in which leakage of information on secrets occurs as a result of performing a *computation* on secrets. Some well-known examples of such attacks include Kocher's timing attacks [Koc96] and power attacks [KJJ99]. A basic defining feature of a computational side-channel attack, as put forth by [MR04] is that *computation and only computation leaks information*. Namely, portions of memory which are not involved in computation do not leak information. A growing number of works [MR04, ISW03, PSP$^+$08, GKR08, DP08] have proposed cryptographic algorithms provably robust against computational side-channel attacks, by limiting in various ways the portions of the secret key which are involved in each step of the computation.

In the work on *one time programs* this is taken to an extreme [GKR08]. Goldwasser, Tauman-Kalai, and Rothblum show how by using a new proposed type of secure-memory which never touches any secrets or data which is not ultimately fully revealed, it is possible to perform any secure computations which is provably secure against *all* computational side channel attacks.

Memory-attacks proposed by Akavia, Goldwasser, and Vaikuntanathan [AGV09] are an entirely very different family of side-channel attacks that are not included in the computational side-channel attack family, as they violate the basic premise of [MR04] that *only computation* leaks information. This class of attacks was inspired by (although not restricted to) the memory-freezing attack introduced recently by Halderman et al. [HSH$^+$08], where its is shown how to measure a significant fraction of the bits of secret keys if the keys were *ever stored* in a part of memory (e.g.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 369–370, 2009.

DRAM), which could be accessed by an adversary even after the power of the machine has been turned off. Thus, information leaks about portions of the secret key which may have never been involved in any computation. A memory-attack leaks a bounded number of bits computed as a result of applying *an arbitrary function* of bounded length (smaller than than the size of the secret key) to the content of the secret key of a cryptographic algorithm. Naturally, this family of attacks is inherently parameterized and quantitative in nature, as if the attack would uncover the entire secret key at the outset, there would be no hope for any cryptography. The work of [AGV09] exhibits a public-key encryption algorithm which is especially robust against memory-attacks. Its security is based on the computationally intractability of the *learning with errors* (LWE) problem which is related to the intractability of approximating the length of the shortest vector in an integer lattice. Finally, a new interesting variant on the idea of memory attacks, had been proposed by Tauman-Kalai etal [DTKL09] in their work on security with auximlary-inputs. They propose to replace the restriction of revealing a length shrinking function of the secret, to revealing functions of the secret which are exponentially hard to invert.

In this talk we will survery this development, with special emphasis on the works of [GKR08, AGV09, DTKL09].

References

[AGV09] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simulatneous hard-core bits and cryptography against memory attack. In: TCC (2009)

[DP08] Dziembowski, S., Pietrzak, K.: Leakage-resilient stream ciphers. In: The IEEE Foundations of Computer Science (2008)

[DTKL09] Dodis, Y., Tauman-Kalai, Y., Lovett, S.: Cryptography with auxilary input. In: STOC (2009)

[GKR08] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)

[HSH+08] Halderman, A., Schoen, S., Heninger, N., Clarkson, W., Paul, W., Calandrino, J., Feldman, A., Appelbaum, J., Felten, E.: Lest we remember: Cold boot attacks on encryption keys. In: Usenix Security Symposium (2008)

[ISW03] Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)

[KJJ99] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

[Koc96] Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)

[MR04] Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)

[PSP+08] Petit, C., Standaert, F.-X., Pereira, O., Malkin, T., Yung, M.: A block cipher based pseudo random number generator secure against side-channel key recovery. In: ASIACCS, pp. 56–65 (2008)

Salvaging Merkle-Damgård for Practical Applications

Yevgeniy Dodis[1], Thomas Ristenpart[2], and Thomas Shrimpton[3]

[1] Dept. of Computer Science, New York University
dodis@cs.nyu.edu
http://www.cs.nyu.edu/~dodis/
[2] Dept. of Computer Science & Engineering, University of California San Diego
tristenp@cs.ucsd.edu
http://www-cse.ucsd.edu/users/tristenp
[3] Dept. of Computer Science, Portland State University
Faculty of Informatics, University of Lugano
teshrim@cs.pdx.edu
http://www.cs.pdx.edu/~teshrim

Abstract. Many cryptographic applications of hash functions are analyzed in the random oracle model. Unfortunately, most concrete hash functions, including the SHA family, use the iterative (strengthened) Merkle-Damgård transform applied to a corresponding compression function. Moreover, it is well known that the resulting "structured" hash function cannot be generically used as a random oracle, even if the compression function is assumed to be ideal. This leaves a large disconnect between theory and practice: although no attack is known for many concrete applications utilizing existing (Merkle-Damgård based) hash functions, there is no security guarantee either, even by idealizing the compression function.

Motivated by this question, we initiate a rigorous and modular study of developing new notions of (still idealized) hash functions which would be (a) natural and elegant; (b) sufficient for arguing security of important applications; and (c) provably met by the (strengthened) Merkle-Damgård transform, applied to a "strong enough" compression function. In particular, we develop two such notions satisfying (a)-(c): a *preimage aware* function ensures that the attacker cannot produce a "useful" output of the function without already "knowing" the corresponding preimage, and a *public-use random oracle*, which is a random oracle that reveals to attackers messages queried by honest parties.

1 Introduction

The primary security goal for cryptographic hash functions has historically been collision-resistance. Consequently, in-use hash functions, such as the SHA family of functions [28], were designed using the (strengthened[1]) Merkle-Damgård

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

[1] We do not mean to imply that there is a *weak* MD transform, but this name seems to be in common use.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 371–388, 2009.

(MD) transform [27,17]: the input message M is suffix-free encoded (e.g. by appending a message block containing the length of M) and then digested by the cascade construction using an underlying fixed-input-length (FIL) compression function. The key security feature of the strengthened MD transformation is that it is *collision-resistance preserving* [17,27]. Namely, as long as the FIL compression function is collision-resistant, the resulting variable-input-length (VIL) hash function will be collision-resistant too.

Random oracle model. Unfortunately, the community has come to understand that collision-resistance alone is insufficient to argue the security of many important applications of hash functions. Moreover, many of these applications, like Fiat-Shamir [22] signatures or RSA [4] encryption, are such that no standard model security assumption about the hash function appears to suffice for proving security. On the other hand, no realistic attacks against these applications have been found. Motivated in part by these considerations, Bellare and Rogaway [4] introduced the Random Oracle (RO) model, which models the hash function as a public oracle implementing a random function. Using this abstraction, Bellare and Rogaway [4,5,6] and literally thousands of subsequent works managed to formally argue the security of important schemes. Despite the fact that a proof in the RO model does *not* always guarantee security when one uses a real (standard model) hash function [13], such a proof does provide evidence that the scheme is structurally sound.

Is Merkle-Damgård a good design? Given the ubiquity of MD-based hash functions in practice, and the success of the RO model in provable security, it is natural to wonder if a MD-based hash function H is reasonably modeled as a RO, at least when the compression function is assumed to be ideal. But even without formalizing this question, one can see that the answer is negative. For example, the well-known *extension attack* allows one to take a value $H(x)$ for *unknown* x, and then compute the value $H(x, \langle \ell \rangle, y)$, where ℓ is the length of x and y is an arbitrary suffix. Clearly, this should be impossible for a truly random function. In fact, this discrepancy leads to simple attacks for *natural* schemes proven secure in the random oracle model (see [16]).

Consequently, Coron et al. [16] adapted the indifferentiability framework of Maurer et al. [26] to define formally what it means to build a secure VIL-RO from smaller (FIL) idealized components (such as an ideal compression function or ideal cipher). Not surprisingly, they showed that the strengthened MD transform does not meet this notion of security, even when applied to an ideal compression function. Although [16] (and several subsequent works [2,3,25]) presented straightforward fixes to the MD paradigm that yield hash functions indifferentiable from a VIL-RO, we are still faced with a large disconnect between theory and practice. Namely, many applications only enjoy proofs of security when the hash function is modeled as a "monolithic" VIL-RO, while in practice these applications use existing MD-based hash functions which (as we just argued) are demonstrably differentiable from a monolithic RO (even when compression

functions are ideal). Yet despite this gap, *no* practical attacks on the MD-based design (like the extension attack) seem to apply for these important applications.

"Salvaging" Merkle-Damgård. The situation leads us to a question not addressed prior to this work: *given a current scheme that employs an* MD*-based hash function H and yet does not seem vulnerable to extension-type attacks, can we prove its security (at least if the compression function f is assumed to be ideal)*? The most direct way to answer this question would be to re-prove, from scratch, the security of a given application when an MD-based hash function is used. Instead, we take a more modular approach consisting of the following steps:

(1) Identify a natural (idealized) property X that is satisfied by a random oracle.
(2) Argue that X *suffices* for proving the security of a given (class of) application(s), originally proved secure when H is modeled as a monolithic RO.
(3) Argue that the *strengthened MD-transform is property-preserving for* X; that is, as long as the compression function f satisfies X, then the VIL hash H satisfies X.
(4) Conclude that, as long as the compression function f satisfies X, the given (class of) application(s) is secure with an MD-based hash function H.

Although this approach might not be applicable to all scenarios, when it *is* applicable it has several obvious advantages over direct proofs. First, it supports proofs that are easier to derive, understand, and verify. Second, proving that a hash function satisfying X alone is enough (as opposed to being like a "full-blown" RO) for a given application elucidates more precisely which (idealized) property of the hash function is essential for security. Third, if the property X is natural, it is interesting to study in its own right. Indeed, we will show several applications of our notions which are quite general and not necessarily motivated by salvaging the MD transform. Finally, due to point (4), it suffices to argue/assume "only" that the compression function f — a smaller and much-better-studied object — satisfies property X.

So which properties X? We introduce two: *preimage awareness* and indifferentiability from a *public-use random oracle*.

1.1 Preimage Aware Functions

Intuitively, a function being Preimage Aware (PrA) means that if an attacker is able to find a "later useful" output y of the hash function H, then it must "already know" a corresponding preimage x. A bit more precisely, assume we build H using some ideal primitive P (which could be a compression function or a block cipher). Then, if the attacker A produces a value y at some point in time, either one can immediately "extract" the corresponding (unique) preimage x of y from the transcript of calls that A made to P so far, or, if one fails to do so, A is exceedingly unlikely to find a valid preimage of y even with the benefit of additional calls to P. Our notion is very similar in spirit to the notion of plaintext awareness for encryption schemes [4,1] and the notion of extractability for perfectly one-way functions [11,12]; we discuss these further below.

We notice that random oracles are clearly PrA. In fact, preimage awareness precisely captures the spirit behind a common proof technique used in the RO model, often referred to as *extractability*, making it an interesting notion to consider. We also show that preimage awareness is a natural strengthening of collision-resistance (CR). That preimage awareness lies between being a RO and CR turns out to be quite useful: informally, a PrA function is "strong enough" to be a good replacement for a RO in some applications (where CR is insufficient), and yet the notion of preimage awareness is "weak enough" to be preserved by strengthened MD (like CR).

Merkle-Damgård preserves preimage awareness. We show that the *(strengthened) MD transform preserves preimage awareness*, in stark contrast to the fact that it does *not* preserve indifferentiability from a RO [16]. Thus, to design a variable-input-length preimage aware (VIL-PrA) function, it is sufficient to construct a FIL-PrA function, or, even better, argue that existing compression functions are PrA, *even when they are not necessarily (indifferentiable from) random oracles*. The proof of this is somewhat similar to (but more involved than) the corresponding proof that MD preserves collision-resistance.

Application: domain extension for ROs. A PrA hash function is *exactly* what is needed to argue secure domain extension of a random oracle. More precisely, assuming h is a FIL-RO, and H is a VIL-PrA hash function (whose output length matches that of the input of f), then $F(x) = h(H(x))$ is indifferentiable from a VIL-RO. Ironically, when H is just CR, the above construction of F was used by [16] to argue that CR functions are not sufficient for domain extension of a RO. Thus, the notion of PrA can be viewed simultaneously as a non-trivial strengthening of CR, which makes such domain extension work, while also a non-trivial weakening of RO, which makes it more readily achieved.

Recipe for hash design. The previous two properties of PrA functions give a general recipe for how to construct hash functions suitable for modeling as a VIL-RO. First, invest as must as needed to construct a strong FIL function h (i.e. one suitable for modeling as a FIL-RO.) Even if h is not particularly efficient, this is perhaps acceptable because it will only be called once per message (on a short input). Second, specify an efficient construction of a VIL-PrA hash function built from some cryptographic primitive P. But for this we use the fact that MD is PrA-preserving; hence, it is sufficient to focus on constructing a FIL-PrA compression function f from P, and this latter task could be much easier than building from P an object suitably like a FIL-RO.

Adopting our more modular point-of-view, several existing hash constructions in the literature [16,2,3,30,19] enjoy an easier analysis. For example, the NMAC construction of [16] becomes an example of our approach, where the outer h and the inner f are both implemented to be like (independent) FIL-ROs. In [16] it is argued directly, via a difficult and long argument, that the inner f can be replaced by the Davies-Meyer construction (in the ideal-cipher model), despite the fact that Davies-Meyer is *not* itself indifferentiable from a FIL-RO. We can

instead just prove that Davies-Meyer is PrA (which requires only a few lines due to the existing proof of CR [7]) and then conclude.

Lifting from CR to PrA. Another important aspect of preimage awareness is that, for many important constructions, it gives a much more satisfactory security target than collision resistance. Indeed, there exists a large body of work [29,7,23,24,32,33,31,19] building FIL-CR hash functions from idealized block ciphers and permutations. On the one hand, it seems very hard to prove the security of such schemes in the standard model, since there exists a black-box separation [34] between collision-resistant hash functions and standard-model block ciphers (which are equivalent to one-way functions). On the other hand, it seems quite unsatisfactory that one starts with such a "powerful" idealized primitive (say, an ideal cipher), only to end up with a much "weaker" standard model guarantee of collision resistance (which is also insufficient for many applications of hash functions). The notion of preimage awareness provides a useful solution to this predicament. We show that *all* the constructions proven CR in [29,7,33,31,19] are provably PrA. This is interesting in its own right, but also because one can now use these practical constructions within our aforementioned recipe for hash design. We believe (but offer no proof) that most other CR ideal-primitive-based functions, e.g. [23,24,32], are also PrA.

Other applications/connections? We believe that PrA functions have many more applications than the ones so far mentioned. As one example, PrA functions seem potentially useful for achieving straight-line extractability for various primitives, such as commitments or zero-knowledge proofs. These, in turn, could be useful in other contexts. As already mentioned, preimage awareness seems to be quite related to the notion of *plaintext awareness* in public-key encryption schemes [5,1], and it would be interesting to formalize this potential connection. PrA functions are also very related to so called *extractable hash functions* (EXT) recently introduced by Canetti and Dakdouk [11,12]. However, there are some important differences between EXT and PrA, which appear to make our respective results inapplicable to each other: (a) EXT functions are defined in the standard model, while PrA functions in an idealized model; (b) EXT functions are keyed (making them quite different from in-use hash functions), while PrA functions can be keyed or unkeyed; (c) EXT functions do not permit the attacker to sample *any* "unextractable" image y, while PrA functions only exclude images y which could be later "useful" to the attacker; (d) EXT functions allow the extractor to depend on the attacker, while PrA functions insist on a universal extractor.

1.2 Public-Use Random Oracles

Next, we consider applications that never evaluate a hash function on secret data (i.e. data that must be hidden from adversaries). This means that whenever the hash function is evaluated on some input x by an honest party C, it is safe to immediately give x to the attacker A. We model this by formalizing the notion of a *public-use random oracle* (pub-RO); such a RO can be queried by adversaries to

reveal all so-far-queried messages. This model was independently considered, under a different motivation, by Yoneyama et al. [36] using the name leaky random oracle. Both of our papers observe that this weakening of the RO model is actually enough to argue security of many (but, certainly, not all) classical schemes analyzed in the random oracle model. In particular, a vast majority of digital signature schemes, including Full Domain Hash (FDH) [4], probabilistic FDH [15], Fiat-Shamir [22], BLS [10], PSS [6] and many others, are easily seen secure in the pub-RO model. For example, in the FDH signature scheme [4], the RO H is only applied to the message m supplied by the attacker, to ensure that the attacker cannot invert the value $H(m)$ (despite choosing m). Other applications secure in the pub-RO model include several identity-based encryption schemes [9,8], where the random oracle is only used to hash the user identity, which is public.

We go on to formalize this weakening of ROs in the indifferentiability framework of Maurer et al. [26]. This allows us to define what it means for a hash function H (utilizing some ideal primitive P) to be indifferentiable from a *public-use* random oracle (pub-RO). As our main technical result here, we argue that the MD transform preserves indifferentiability from a pub-RO, even though it does not preserve general indifferentiability from a (regular) RO. To get some intuition about this fact, it is instructive to examine the extension attack mentioned earlier, which was the root of the problem with MD for general indifferentiability. There one is worried about adversaries being able to infer the hash output on a message with unknown prefix. In the public-use setting, this is not an issue at all: the security of a public-use application could never be compromised by extension attacks since all messages are known by the attacker.

As a corollary of this result (and the composition theorem of [26]), we'll see that if the compression function f is indifferentiable from a FIL pub-RO, we can immediately give a security proof for the above-mentioned public-use applications. In particular, this is true when f is modeled as an ordinary FIL-RO.In the full version [21], we discuss the more complicated case of the Davies-Meyer compression function.

2 Preliminaries

When S is a set, $x \xleftarrow{\$} S$ means to sample uniformly from S and assign the value to x. When Dom and Rng are non-empty sets, let $\mathsf{RF}_{Dom,Rng}$ be the algorithm that implements a lazily-sampled random oracle mapping from Dom to Rng. We shorten this to $\mathsf{RF}_{Dom,n}$ or $\mathsf{RF}_{N,n}$ when the range (resp.) domain and range are bit strings of fixed lengths $N, n > 0$; $\mathsf{RF}_{*,\tau}$ is a random oracle with constant output stretch τ. Let $\kappa, n > 0$ be integers. A block cipher is a map $E\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ such that $E(k,\cdot)$ is a permutation for all $k \in \{0,1\}^\kappa$. Let $\mathsf{BC}(\kappa, n)$ be the set of all such block ciphers.

For any algorithm f that accepts inputs from $Dom \subseteq \{0,1\}^*$, we write $\mathsf{Time}(f,m)$ to mean the maximum time to run $f(x)$ for any input $x \in \{0,1\}^m \subseteq Dom$. When f is a function with domain $Dom \subseteq \{0,1\}^*$, we define $\mathsf{Time}(f,m)$ to be the minimum, over all programs T_f that implement the mapping f, of the size of T_f plus the worst case running time of T_f over all elements $x \in \{0,1\}^m \subseteq Dom$. In

either case, when we suppress the second argument, writing just $\mathsf{Time}(f)$, we mean to maximize over all strings in the domain. Running times are relative to some fixed underlying RAM model of computation, which we do not specify here.

As a small abuse of standard notation, we write $\mathcal{O}(X)$ to hide fixed, absolute constants that are much smaller than the argument X.

Interactive TMs. An Interactive Turing Machine (ITM) accepts inputs via an input tape, performs some local computation using internal state that persists across invocations, and replies via an output tape. An ITM might implement various distinct functionalities $f_1, f_2, \ldots$ that are to be exposed to callers. We write $P = (f_1, f_2, \ldots)$ for an ITM implementing $f_1, f_2, \cdots$. When functionalities f_i, f_j (say) do not share state, we say that f_i and f_j are *independent* functionalities; these will be explicitly noted. We write M^P if an ITM M has access to all interfaces of P and write M^{f_i} if M has access only to a particular interface f_i of P. We sometimes use the moniker *ideal primitive* to refer to an ITM; this is to emphasize the use of an ITM as building block for some larger functionality. We write $\mathcal{F} = \mathsf{RF}_{Dom,Rng}$ to signify that $\mathcal{F}$ is the ideal primitive that implements the algorithm $\mathsf{RF}_{Dom,Rng}$.

Hash Functions and Merkle-Damgård. Let $Dom \subseteq \{0,1\}^*$ be a non-empty set of strings, and Rng be a non-empty set (typically $\{0,1\}^n$ for some integer $n > 0$). A *hash function* is then a map $H\colon Dom \to Rng$. We will be concerned with hash functions that use (oracle access to) an underlying ideal primitive P. We write H^P when we want to make this dependency explicit. When the primitive is clear from context, we will sometimes suppress reference to it. When computing $\mathsf{Time}(H, \cdot)$, calls to P are unit cost. Similar to our definition of $\mathsf{Time}(H, m)$, we write $\mathsf{NumQueries}(H, m)$ for the minimum, over all programs T_H that compute H, of the maximum number of queries to P required to compute $H^P(x)$ for any $x \in \{0,1\}^m \subseteq Dom$.

The primary method by which hash functions are constructed from underlying primitives is the strengthened Merkle-Damgård transform (SMD). For integers $n, d > 0$, let $f^P\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$ be a *compression function* (using idealized primitive P). Let $y_0 = IV$, a fixed constant. We write $\mathrm{SMD}[f^P]$ for the algorithm that works on input $M \in \{0,1\}^*$ by first computing $m_1 \cdots m_\ell \leftarrow \mathsf{sfpad}(M)$, running $y_i \leftarrow f^P(y_{i-1}, m_i)$ for each $i \in [1\,..\,\ell]$ and returning y_ℓ. Here $\mathsf{sfpad}(\cdot)$ is a function that returns a *suffix-free encoding* of M that is parsed into ℓ blocks of d-bits each, where $\ell \geq \lceil |M|/d \rceil$ is defined by the encoding. A suffix-free encoding has the property that for any M, M' such that $|M| < |M'|$ the string returned by $\mathsf{sfpad}(M)$ is not a suffix of $\mathsf{sfpad}(M')$. (For example, appending to a message its length.) Similarly, we write $\mathrm{MD}[f^P]$ for the algorithm that splits input $M \in (\{0,1\}^d)^+$ into blocks $m_1, \cdots, m_\ell$, each of size d bits, then runs $y_i \leftarrow f^P(y_{i-1}, m_i)$ for each $i \in [1\,..\,\ell]$, and returns y_ℓ. When the underlying compression function is itself an idealized primitive provided as an oracle, the algorithms SMD and MD work in the obvious manner.

Pseudorandom oracles. Following [16], we utilize the indifferentiability framework [26] to formalize the notion of "behaving like a random oracle", which

we call being a pseudorandom oracle (PRO) following [2,3]. Fix non-empty sets Dom, Rng. Let P be an ideal primitive and let $\mathcal{F} = \mathsf{RF}_{Dom,Rng}$. A simulator $\mathcal{S}$ is an ITM that matches the number of interfaces of P and has oracle access to $\mathcal{F}$. A PRO adversary A has access to oracles and outputs a bit. We define the pro advantage of a PRO adversary A against a function H^P mapping from Dom to Rng by

$$\mathbf{Adv}^{\mathrm{pro}}_{H,\mathcal{S}}(A) = \Pr\left[\, A^{H,P} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathcal{F},\mathcal{S}} \Rightarrow 1 \,\right]$$

where the first probability is over the coins used by A and primitive P, and the second is over the coins used by A, $\mathcal{F}$, and $\mathcal{S}$. Note that a crucial aspect of the definition is that the simulator, while able to query $\mathcal{F}$ itself, does *not* get to see the queries made by the adversary to $\mathcal{F}$.

3 Preimage Awareness

Suppose H is a hash function built from an (ideal) primitive P. We seek to, roughly speaking, capture a notion which states that an adversary who knows a "later useful" output z of H^P must "already know" (be aware of) a particular corresponding preimage x. We can capture the spirit of this notion using a deterministic algorithm called an *extractor*. Consider the following experiment. An adversary A initially outputs a range point z. The extractor is run on two inputs: z and an *advice string* α. The latter contains a description of all of A's queries so far to P and the corresponding responses. The extractor outputs a value x in the domain of H. Then A runs again and attempts to output a preimage x' such that $H^P(x) = z$ but $x \neq x'$. Informally speaking, if no adversary can do so with high probability, then we consider H to be preimage aware. We now turn to formalizing a notion based on this intuition, but which allows multiple, adaptive attempts by the adversary to fool the extractor.

Fix sets $Dom \subseteq \{0,1\}^*$ and Rng, and let A be an adversary that outputs a string $x \in Dom$. In the *preimage awareness* (pra) experiment defined in Figure 1, the adversary is provided with two oracles. First, an oracle P that provides access to the (ideal) primitive P, but which also records all the queries and their responses in an advice string α. (We assume that when P is providing an interface to multiple primitives, it is clear from the advice string to which primitive each query was made.) Second, an *extraction oracle* Ex. The extraction oracle provides an interface to an *extractor* $\mathcal{E}$, which is a deterministic algorithm that takes as input a point $z \in Rng$ and the advice string α, and returns a point in $Dom \cup \{\perp\}$.

For hash function H, adversary A, and extractor $\mathcal{E}$, we define the advantage relation

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A) = \Pr\left[\, \mathbf{Exp}^{\mathrm{pra}}_{H,\mathcal{E},A} \Rightarrow \mathsf{true} \,\right]$$

where the probabilities are over the coins used in running the experiments. We will assume that an adversary never asks a query outside of the domain of the queried oracle. We use the convention that the running time of the adversary A

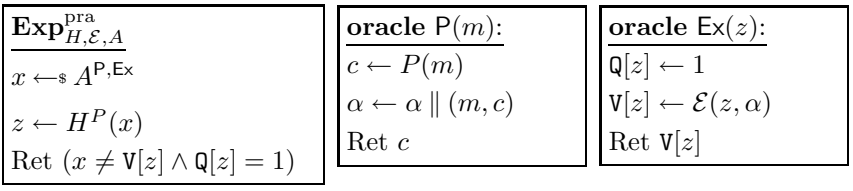


Fig. 1. (Left) Experiment for defining preimage awareness (PrA) for hash function H, extractor $\mathcal{E}$ and adversary A. **(Right)** Description of the oracles used in the PrA experiment extractor $\mathcal{E}$. The (initially empty) advice string α, the (initially empty) array $\mathtt{V}$, and the (initially everywhere $\perp$) array $\mathtt{Q}$ are global.

does not include the time to answer its queries (i.e. queries are unit cost). When there exists an efficient extractor $\mathcal{E}$ such that $\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A)$ is small for all reasonable adversaries A, we say that the hash function H is preimage aware (PrA). (Here "efficient", "small", and "reasonable" are meant informally.)

Remarks. As mentioned, the above formalization allows multiple, adaptive challenge queries to the extraction oracle. This notion turned out to be most convenient in applications. One can instead restrict the above notion to a single query (or to not allow adaptivity) resulting in a definition with slightly simpler mechanics. In the full version [21] we discuss such alternative formulations of preimage awareness.

4 Relationships between PrA, CR, and Random Oracles

Our new notion preimage awareness is an interesting middle point in the continuum between objects that are CR (on one end) and those that are random oracles (on the other). More formally speaking, we'll see in a moment that preimage awareness is a strictly stronger notion than CR while it is easy to see that it is a strictly weaker notion than indifferentiability from a random oracle. This is interesting for several reasons. First, we show that a PrA function is a secure domain extender for fixed-input-length random oracles, unlike CR functions [16]. (This already suggests that CR does not necessarily imply PrA.). Preimage awareness is consequently a very useful strengthening of CR, not to mention that it provides rigor to the folklore intuition that CR functions are insufficient for this application due to a lack of extractability. Second, the MD transform preserves preimage awareness. This is in stark contrast to the fact that MD (even if one uses strengthening) does *not* preserve indifferentiability from a random oracle (i.e. PRO-Pr). In the rest of this section, we explore these facts in more detail.

One can view preimage awareness as a strengthening of collision resistance in the following way. Say that queries to P allow the adversary to compute distinct domain points x, x' such that $H^P(x) = H^P(x') = z$. The adversary can make an extraction query on z, and then succeed in the PrA game by returning whichever of x and x' is not extracted from (z, α) by the extractor.

On the other hand, it is not hard to see that a RO is a PrA function. (consider an extractor that simply scans the advice string looking for the challenge point z). We now turn to the two claims mentioned above. The next theorem captures that any PrA function is a good domain extender for an FIL random oracle. The proof is given in the full version [21].

Theorem 1. [RO domain extension via PrA] Let P be an ideal primitive and $H^P\colon \mathit{Dom} \to \mathit{Rng}$ be a hash function. Let R be an ideal primitive with two interfaces that implements independent functionalities P and $\mathcal{R} = \mathsf{RF}_{\mathit{Rng},\mathit{Rng}}$. Define $F^R(M) = \mathcal{R}(H^P(M))$. Let $\mathcal{F} = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$. Let $\mathcal{E}$ be an arbitrary extractor for H. Then there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PRO adversary A making at most (q_0, q_1, q_2) queries to its three oracle interfaces, there exists a PrA adversary B such that

$$\mathbf{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(A) \leq \mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(B)\ .$$

Simulator $\mathcal{S}$ runs in time $\mathcal{O}(q_1 + q_2 \cdot \mathsf{Time}(\mathcal{E}))$. Let ℓ_{max} the the length (in bits) of the longest query made by A to it's first oracle. Adversary B runs in time that of A plus $\mathcal{O}(q_0 \cdot \mathsf{Time}(H, \ell_{max}) + q_1 + q_2)$, makes $q_1 + q_0 \cdot \mathsf{NumQueries}(H, \ell_{max})$ primitive queries, q_2 extraction queries, and outputs a preimage of length at most ℓ_{max}. □

Theorem 1 shows that preimage awareness is a strong enough notion to provide secure domain extension for random oracles. At the same time, the next theorem shows that it is "weak" enough to be preserved by SMD. We consider SMD based on any suffix-free padding function $\mathsf{sfpad}\colon \{0,1\}^* \to (\{0,1\}^d)^+$ that is injective. Further we assume it is easy to strip padding, namely that there exists an efficiently computable function $\mathsf{unpad}\colon (\{0,1\}^d)^+ \to \{0,1\}^* \cup \{\perp\}$ such that $x = \mathsf{unpad}(\mathsf{sfpad}(x))$ for all $x \in \{0,1\}^*$. Inputs to unpad that are not valid outputs of sfpad are mapped to $\perp$ by unpad.

Theorem 2. [SMD is PrA-preserving] Fix $n, d > 0$ and let P be an ideal primitive. Let $h^P\colon \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function, and let $H = \mathrm{SMD}[h^P]$. Let $\mathcal{E}_h$ be an arbitrary extractor for the PrA-experiment involving h. Then there exists an extractor $\mathcal{E}_H$ such that for all adversaries A making at most q_p primitive queries and q_e extraction queries and outputting a message of at most $\ell_{max} \geq 1$ blocks there exists an adversary B such that

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}_H}(A) \leq \mathbf{Adv}^{\mathrm{pra}}_{h,\mathcal{E}_h}(B)\ .$$

$\mathcal{E}_H$ runs in time at most $\ell_{max}\,(\mathsf{Time}(\mathcal{E}_h) + \mathsf{Time}(\mathsf{unpad}))$. B runs in time at most that of A plus $\mathcal{O}(q_e \ell_{max})$, makes at most $\ell_{max} \cdot \mathsf{NumQueries}(h, \ell_{max}) + q_p$, and makes at most $q_e \ell_{max}$ extraction queries. □

Proof. We start by defining the adversary B; the extractor $\mathcal{E}_H$ is implicit in its description.

adversary $B^{\mathsf{P},\mathsf{Ex}}(\varepsilon)$:

$x^* \leftarrow\!\!\$\ A^{\mathsf{P},\mathsf{SimEx}}$
$x^*_\ell \cdots x^*_1 \xleftarrow{d} \mathsf{sfpad}(x^*)$; $c^*_{\ell+1} \leftarrow IV$
For $i = \ell$ down to 1 do
  $c^*_i \leftarrow h^{\mathsf{P}}(c^*_{i+1} \,\|\, x^*_i)$
  If $\mathtt{Q}[c^*_i] = 1$ and $\mathtt{E}[c^*_i] \neq c^*_{i+1} \,\|\, x^*_i$ then
    Ret $c^*_{i+1} \,\|\, x^*_i$
Ret $\perp$

subroutine $\mathsf{SimEx}(z, \alpha)$:

$i \leftarrow 1$; $c_1 \leftarrow z$
While $i \leq \ell_{max}$ do
  $c_{i+1} \,\|\, x_i \leftarrow \mathsf{Ex}(c_i, \alpha)$
  $\mathtt{Q}[c_i] \leftarrow 1$; $\mathtt{E}[c_i] \leftarrow c_{i+1} \,\|\, x_i$
  If $c_{i+1} = \perp$ then Ret $\perp$
  $x \leftarrow \mathsf{unpad}(x_i \cdots x_1)$
  If $c_{i+1} = IV$ and $x \neq \perp$ then
    Ret x
  $i \leftarrow i + 1$
Ret $\perp$

Adversary B answers A's primitive queries by forwarding to its own oracle P. It answers A's extraction queries using the subroutine SimEx (which makes use of B's extraction oracle). The code $c_{i+1} \,\|\, x_i \leftarrow \mathsf{Ex}(c_i, \alpha)$ means take the string returned from the query and parse it into an n-bit string c_{i+1} and a d-bit string x_i. If the oracle returns $\perp$, then c_{i+1} and x_i are both assigned $\perp$. The code $x^*_\ell \cdots x^*_1 \xleftarrow{d} \mathsf{sfpad}(x^*)$ means take the output of $\mathsf{sfpad}(x^*)$ and parse it into ℓ d-bit blocks $x^*_\ell, \ldots, x^*_1$. The tables $\mathtt{Q}$ and $\mathtt{E}$, which record if a value was queried to Ex and the value returned by the query, are initially everywhere $\perp$.

The extractor $\mathcal{E}_H$ works exactly the same as the code of SimEx except that queries to Ex are replaced by directly running $\mathcal{E}_h$ and the tables $\mathtt{Q}$ and $\mathtt{E}$ can be omitted. Loosely, extractor $\mathcal{E}_H$, when queried on a challenge image z, uses $\mathcal{E}_h$ to compute (backwards) the preimages of each iteration of h leading to z. When a chaining variable equal to IV is extracted, the function unpad is applied to the extracted message blocks. If it succeeds, then the result is returned.

Note that we reverse the (usual) order of indices for message blocks and chaining variables (starting high and counting down, e.g. $x^*_\ell \cdots x^*_1$) for both the extractor and B due to the extractor working backwards.

To lower bound B's advantage by the advantage of A we first point out that, by construction of $\mathcal{E}_H$, the values returned by the simulated SimEx are distributed identically to the values returned during execution of $\mathbf{Exp}^{\mathrm{pra}}_{H,\mathcal{E}_H,A}$. Thus we have that $\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}_H}(A) = \Pr[x^* \text{ satisfies}]$ where the event "x^* satisfies", defined over the experiment $\mathbf{Exp}^{\mathrm{pra}}_{h,\mathcal{E}_B,B}$, occurs when the message x^* satisfies the conditions of winning for A. Namely that $H^P(x^*)$ was queried to SimEx and the reply given was not equal to x^*. We call x^* a satisfying preimage for A. We will show that whenever x^* is a satisfying preimage for A, with $x^*_\ell \cdots x^*_1 \xleftarrow{d} \mathsf{sfpad}(x^*)$, there exists a k with $1 \leq k \leq \ell$ for which adversary B returns the string $c^*_{k+1} \,\|\, x^*_k$ and this string is a satisfying preimage for B (i.e. one that wins the PrA experiment against h for B). This will establish that

$$\Pr[\, x^* \text{ satisfies} \,] \leq \mathbf{Adv}^{\mathrm{pra}}_{h,\mathcal{E}_h}(B) \,. \tag{1}$$

Consider the query $\mathsf{SimEx}(H^P(x^*))$ necessarily made by A. Let $c_{j+1} \| x_j, \ldots, c_2 \| x_1$ be the sequence of values returned by the Ex queries made by SimEx in the course of responding to A's query. Necessarily $1 \leq j \leq \ell_{max}$ and $1 \leq \ell \leq \ell_{max}$.

We will show that there exists a k such that $1 \leq k \leq \min\{j, \ell\}$ and $c_{k+1} \,\|\, x_k \neq c^*_{k+1} \,\|\, x^*_k$. (This includes the possibility that $c_{k+1} = \perp$ and $x_k = \perp$.) First we use this fact to conclude. Since $k \leq j$ it means that c_k was queried to Ex. If $c_k = c^*_k = H^P(c^*_{k+1} \,\|\, x^*_k)$ we are done, because then $c^*_{k+1} \,\|\, x^*_k$ is a satisfying preimage for B. Otherwise, $c_k \neq c^*_k$ and we can repeat the reasoning for $k-1$. At $k = 1$ we have that, necessarily, $c_k = c^*_k$ since this was the image queried by A. Thus there must exist a satisfying preimage, justifying (1).

We return to showing the existence of k such that $1 \leq k \leq \min\{j, \ell\}$ and $c_{k+1} \,\|\, x_k \neq c^*_{k+1} \,\|\, x^*_k$. Assume for contradiction that no such k exists, meaning that $c^*_{i+1} \,\|\, x^*_i = c_{i+1} \,\|\, x_i$ for $1 \leq i \leq \min\{j, \ell\}$. If $j > \ell$, then since $c_j = IV$ and $x^*_\ell \cdots x^*_1 = x_\ell \cdots x_1$ we have a contradiction because in such a situation the loop in SimEx would have halted at iteration ℓ. If $j = \ell$, then having $x^*_\ell \cdots x^*_1 = x_\ell \cdots x_1$ and $c_{\ell+1} = c^*_{\ell+1} = IV$ would imply that SimEx returned $x = x^*$, contradicting that x^* is a satisfying preimage for A. If $j < \ell$, then the loop in SimEx must have stopped iterating because $c_{j+1} = IV$ (if $c_{j+1} = \perp$ we would already have contradicted our assumption regarding k) and $x \neq \perp$. But by assumption we have that $x^*_j \cdots x^*_1 = x_j \cdots x_1$ and so there exist two strings x and x^* for which $\mathsf{sfpad}(x)$ is a suffix of $\mathsf{sfpad}(x^*)$. This contradicts that sfpad provides a suffix-free encoding. ∎

Recall that if a compression function h is both CR and hard to invert for range point the IV, then the plain MD iteration of h is a CR function [17,20]. We prove an analogous theorem for plain MD and preimage awareness in [21]. This is particularly useful in our context, because for the compression functions we will consider (e.g. a FIL random oracle or an ideal cipher based compression function) it is easy to verify that it is difficult to invert a fixed range point. Note that this extra property on h (difficulty of inverting IV) is, in fact, *necessary* for plain MD to provide preimage awareness.

5 Applying Preimage Awareness

The results of the previous section allow us to more elegantly and modularly prove that a hash function construction is a pseudorandom oracle (PRO). Particularly, Theorems 1 and 2 mean that the task of building a PRO is reduced to the significantly simpler task of building a compression function that is PrA. For example, in the case that the compression function is itself suitable to model as an FIL-RO, then it is trivially PrA and so one is finished. However, even if the compression function has some non-trivial structure, such as when based on a block cipher, it is still (relatively) straightforward to prove (suitable compression functions) are PrA. In the rest of this section we show that most CR functions built from an ideal primitive are, in fact, PrA.

Are there applications of preimage awareness beyond analysis of hash functions? We believe the answer is yes. For example, one might explore applications of CR functions, instead analyzing these applications assuming a PrA function. (As one potential application, the CR-function using statistically hiding

commitment scheme of [18] conceivably achieves straight-line extractability given instead a PrA function.) We leave such explorations to future work.

PrA FOR CR CONSTRUCTIONS. There is a long line of research [29,7,23,24], [32,33,31,19] on building compression functions (or full hash functions) that are provably collision-resistant in some idealized model, e.g. the ideal cipher model. We show that in most cases one can generalize these results to showing the constructions are also PrA. We start by showing that the Davies-Meyer and other so-called "type-1" PGV compression functions [29,7] are not only CR but PrA. We also give bounds on the PrA-security of the Shrimpton-Stam compression function [33] (see Theorem 4) and the first two steps of the MCM construction [30] (see Theorem 5); previously these were only known to be CR.

Let us begin by examining the Davies-Meyer compression function, which is defined by $\mathsf{DM}^E(c, m) = E_m(c) \oplus c$ where E is a block cipher. This compression function and the rest of the 12 "type-1" PGV [29] block cipher-based compression functions were proved to be collision-resistant (to the birthday bound) in the ideal-cipher model in [7]. We leverage their results in the proof of the following theorem, given in the full version [21] where results for the other "type-1" functions also appear.

Theorem 3. [Davies-Meyer is PrA.] Fix $\kappa, n > 0$, let $E \leftarrow\!\!\$\ \mathsf{BC}(\kappa, n)$. Let P be an oracle providing an interface to E and E^{-1}. Let $H^P(c, m) = \mathsf{DM}^E(c, m)$. There exists an extractor $\mathcal{E}$ such that for any adversary A making at most q_p queries to P and q_e extraction queries we have

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A) \leq \frac{q_e q_p}{2^{n-1}} + \frac{q_p(q_p+1)}{2^n}$$

where $\mathcal{E}$ runs in time at most $\mathcal{O}(q_p)$. □

Next we show that it is possible to build a PrA compression function from *non-compressing* random functions[2]. In particular, we examine the compression function recently designed by Shrimpton and Stam [33]. They proved that this compression function is nearly optimally collision resistant (i.e. to the birthday bound), and we will now show that it is also PrA. The proof of the following is in [21].

Theorem 4. [Shrimpton-Stam is PrA] Fix $n > 0$. Let $P = (f_1, f_2, f_3)$ be an ideal primitive providing interfaces to independent functionalities $f_1 = \mathsf{RF}_{n,n}$, $f_2 = \mathsf{RF}_{n,n}$ and $f_3 = \mathsf{RF}_{n,n}$. Define a compression function $H^P(c, m) = f_3(f_1(m) \oplus f_2(c)) \oplus f_1(m)$. Then there exists an extractor $\mathcal{E}$ such that for any adversary A making q_p queries to each of f_1, f_2, f_3 (via P) and q_e extraction query, we have

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A) = \mathcal{O}(q_e q_p^2 / 2^n)$$

where the extractor runs in time $\mathcal{O}(q_p^2)$. □

[2] One can view a block cipher as a compressing primitive, since it takes $k+n$ bits and produces n bits.

Dodis et al. [19] also offer a compression function from non-compressing primitives, this being $f(c, m) = f_1(c) \oplus f_2(m)$. A straightforward extension of the argument in [19] shows that this function is PrA for ideal f_1 and f_2. See [21].

Finally, we show that the "mix-compress" portion of the "mix-compress-mix" construction from [30] is PrA as long as the compress step is CR and relatively balanced. First we must define a measure of balance. Associated to any function $F\colon \{0,1\}^* \to \{0,1\}^n$ is the set $\mathsf{Prelm}_F(\ell, z) = \{y \mid y \in \{0,1\}^* \wedge |y| = \ell \wedge F(y) = z\}$ for all $\ell > 0$ and $z \in \{0,1\}^n$. That is, $\mathsf{Prelm}_F(\ell, z)$ contains the length ℓ preimages of z under F. We also define the function $\delta_F(\ell, z) = |(|\mathsf{Prelm}_F(\ell, z)| - 2^{\ell-n})/2^{\ell}|$ related to F. The δ_F function measures how far a particular preimage set deviates from the case in which F is regular. Let $\Delta_F = \max\{\delta_F(\ell, z)\}$, where the maximum is taken over all choices of ℓ and z. The proof of the following is given in [21].

Theorem 5. [Mix-Compress is PrA.] Fix $\tau, n > 0$, let $F\colon \{0,1\}^* \to \{0,1\}^n$ and let P be an ideal primitive implementing $\mathsf{RF}_{*,\tau}$. Let $H^P(m) = F(P(m))$. Let A be a PrA adversary making q_p primitive queries and q_e extraction queries. Then there exists a CR adversary B and an extractor $\mathcal{E}$ such that

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A) \leq q_e q_p\left(\frac{1}{2^n} + \Delta_F\right) + \mathbf{Adv}^{\mathrm{cr}}_{H,F}(B)$$

$\mathcal{E}$ runs in time at most $\mathcal{O}(q_p)$. B runs in time at most that of A plus $\mathcal{O}(q_p)$. □

6 Indifferentiability for Public-Use Random Oracles

In numerous applications, hash functions are applied only to public messages. Such public-use occurs in most signature schemes (e.g. full-domain-hash [4], probabilistic FDH [15], Fiat-Shamir [22], BLS [10], PSS [6]) and even some encryption schemes (e.g. a variant of Boneh-Franklin IBE [14] and Boneh-Boyen IBE [8]). It is easy to verify that the provable security of such schemes is retained even if all hashed messages are revealed to adversaries. We introduce the notion of a public-use random oracle (pub-RO). This is an ideal primitive that exposes two interfaces: one which performs the usual evaluation of a random oracle on some domain point and a second which reveals all so-far evaluated domain points. All parties have access to the first interface, while access to the latter interface will only be used by adversaries (and simulators).

A wide class of schemes that have proofs of security in the traditional random oracle model can easily be shown secure in this public-use random oracle model. Consider any scheme and security experiment for which all messages queried to a RO can be inferred from an adversary's queries during the experiment. Then one can prove straightforwardly the scheme's security in the pub-RO model, using an existing proof in the full RO model as a "black box". For example, these conditions are met for unforgeability under chosen-message attacks of signature schemes that use the RO on messages and for message privacy of IBE schemes that use the RO on adversarially-chosen identities. All the schemes listed in the previous paragraph (and others) fall into these categories.

The pub-RO model was independently considered by Yoneyama et al. [36] (there called the leaky random oracle model) under different motivation. They directly prove some schemes secure when hash functions are modeled as a *monolithic* pub-RO. They do not analyze the underlying structure of MD-based functions.

We utilize the indifferentiability framework of Maurer et al. [26] to formalize a new notion of security for hash constructions: indifferentiability from a public-use RO, which we will call being a *public-use pseudorandom oracle* (pub-PRO). This new security property is weaker than that of being a PRO. We show that iterating a fixed-input-length public-use random oracle (i.e. a compression function) via MD yields a variable-input-length public-use random oracle. Put another way, MD preserves the property of being a pub-PRO.

PUBLIC-USE ROs. Fix sets Dom, Rng. A public-use random oracle (pub-RO) for domain Dom and range Rng is an ideal primitive $\mathcal{F} = (\mathcal{F}_{eval}, \mathcal{F}_{reveal})$ defined as follows. Let $\rho = \mathsf{RF}_{Dom,Rng}$. The evaluation interface $\mathcal{F}_{eval}$, on input $M \in Dom$, first adds the pair $(M, \rho(M))$ to an initially-empty set $\mathcal{M}$ and then returns $\rho(M)$. The reveal interface $\mathcal{F}_{reveal}$ takes no input and returns $\mathcal{M}$ (suitably encoded into a string). We say that $\mathcal{F}$ is a fixed-input-length (FIL) pub-RO if Dom only includes messages of a single length.

INDIFFERENTIABILITY FROM A PUB-RO. Fix sets Dom, Rng. Let P be an ideal primitive and let $\mathcal{F} = (\mathcal{F}_{eval}, \mathcal{F}_{reveal})$ be a pub-RO for domain Dom and range Rng. Let $\mathcal{S}$ be a simulator with oracle access to (both interfaces of) $\mathcal{F}$. Then we define the pub-pro advantage of an adversary A against a construction H^P by

$$\mathbf{Adv}^{\text{pub-pro}}_{H,\mathcal{S}}(A) = \Pr\left[\, A^{H,P} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathcal{F}_{eval},\mathcal{S}} \Rightarrow 1 \,\right]$$

where the first probability is over the coins used by A and primitive P, and the second is over the coins used by A, $\mathcal{F}$, and $\mathcal{S}$. In the second probability experiment, while A has access only to $\mathcal{F}_{eval}$, the simulator $\mathcal{S}$ has oracle access to both interfaces of $\mathcal{F}$. The simulator's ability to call $\mathcal{F}_{reveal}$, thereby seeing all queries so-far-made by A to $\mathcal{F}_{eval}$, is the crucial difference between pub-PRO and PRO.

The composition theorem in [26] (recast to use ITMs in [16]) can be applied to pub-PROs. That is, a cryptographic scheme using a pub-PRO hash construction H^P for some ideal primitive P can have its security analyzed in a setting where H^P is replaced by a monolithic pub-RO $\mathcal{F}$. In this setting, adversaries attacking the scheme can perform queries to $\mathcal{F}_{reveal}$.

MERKLE-DAMGÅRD PRESERVES pub-PRO. Let $f = (f_{eval}, f_{reveal})$ be a FIL pub-RO. Then the next theorem statement, whose proof appears in [21], asserts that MD$[f_{eval}]$ is indifferentiable from a pub-RO. (That SMD$[f_{eval}]$ is a pub-PRO is an immediate corollary.)

Theorem 6. [MD preserves pub-PRO] Fix $n, d > 0$. Let $f = (f_{eval}, f_{reveal})$ be a FIL pub-RO for domain $\{0,1\}^{n+d}$ and range $\{0,1\}^n$. There exists a simulator $\mathcal{S} = (\mathcal{S}_{eval}, \mathcal{S}_{reveal})$ so that for any adversary A

$$\mathbf{Adv}^{\text{pub-pro}}_{\text{MD},\mathcal{S}}(A) \leq \frac{(\sigma q_0 + q_1)^2}{2^n} + \frac{\sigma q_0 + q_1 + 1}{2^n}$$

where q_0 is the maximal number of queries by A to its first oracle, these of length at most σ blocks of d bits, and q_1 is the maximal number of queries by A to either f_{eval} or $\mathcal{S}_{eval}$. Let q_2 be the number of queries by A to either f_{reveal} or $\mathcal{S}_{reveal}$. Then S runs in time that of A plus $\mathcal{O}(q_0\sigma(q_1 + q_2))$ and makes at most $2q_0 + q_0q_1\sigma$ queries. □

Davies-Meyers compression function. One might hope that the Davies-Meyers compression function is pub-PRO analogously to the fact that it is PrA. Unfortunately, this is not the case. Consider the following attack, due to [35]. Let A against $\mathsf{DM}^E(c, x) = E_x(c) \oplus c$ work as follows. It picks a random z and m and then queries its third oracle interface on m, z. When interacting with the pub-RO $\mathcal{F}$ and any simulator $\mathcal{S}$, we see that $\mathcal{S}$ would need to respond with a value c such that $\mathcal{F}_{eval}(c, x) = c \oplus z$. This corresponds to inverting $\mathcal{F}$ on some fixed range point, which is hard. (Note that A has not, before querying the simulator, submitted any queries to $\mathcal{F}$.) Thus the adversary will win easily. On the other hand, in the full version [21], we show that, although DM is not itself pub-PRO, applying MD to it results in a VIL pub-PRO (in the ideal cipher model). We discuss this in more detail in [21].

Acknowledgments

We thank Ilya Mironov, Martijn Stam, and Mihir Bellare for useful discussions regarding this work. We thank Lei Wang for pointing out an error in an earlier version of this work. Yevgeniy Dodis was supported in part by NSF Grants 0831299, 0716690, 0515121, and 0133806. Thomas Ristenpart was supported in part by Mihir Bellare's NSF grants CNS 0524765 and CNS 0627779 and a gift from Intel corporation. He thanks the Faculty of Informatics at the University of Lugano, Switzerland for hosting and supporting him while a portion of this work was done. Thomas Shrimpton was supported by NSF grant CNS 0627752 and SNF grant 200021-122162.

References

1. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
2. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
3. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS 1993, pp. 62–73. ACM Press, New York (1993)
5. Bellare, M., Rogaway, P.: Optimal asymmetric encryption – How to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)

6. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
7. Black, J.A., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–325. Springer, Heidelberg (2002)
8. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
9. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
10. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
11. Canetti, R., Dakdouk, R.R.: Extractable Perfectly One-Way Functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 449–460. Springer, Heidelberg (2008)
12. Canetti, R., Dakdouk, R.: Towards a Theory of Extractable Functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 595–614. Springer, Heidelberg (2009)
13. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
14. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. J. Cryptology (JOC) 20(3), 265–294 (2007)
15. Coron, J.-S.: Optimal Security Proofs for PSS and Other Signature Schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
16. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
17. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
18. Damgård, I.B., Pedersen, T.P., Pfitzmann, B.: On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Sigantures. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 250–265. Springer, Heidelberg (1994)
19. Dodis, Y., Pietrzak, K., Puniya, P.: A New Mode of Operation for Block Ciphers and Length-Preserving MACs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 198–219. Springer, Heidelberg (2008)
20. Dodis, Y., Puniya, P.: Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA? In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 156–173. Springer, Heidelberg (2008)
21. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging Merkle-Damgård for Practical Applications (full version of this paper). IACR ePrint Archive (2009)
22. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
23. Hirose, S.: Provably Secure Double-Block-Length Hash Functions in a Black-Box Model. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 330–342. Springer, Heidelberg (2005)

24. Hirose, S.: Some Plausible Constructions of Double-Block-Length Hash Functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
25. Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
26. Maurer, U.M., Renner, R.S., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
27. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
28. National Institute of Standards and Technology. FIPS PUB 180-1: Secure Hash Standard, Supersedes FIPS PUB 180 (May 11, 1995)
29. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
30. Ristenpart, T., Shrimpton, T.: How to Build a Hash Function from Any Collision-Resistant Function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 147–163. Springer, Heidelberg (2007)
31. Rogaway, P., Steinberger, J.P.: Security/Efficiency Tradeoffs for Permutation-Based Hashing. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008)
32. Rogaway, P., Steinberger, J.P.: Constructing Cryptographic Hash Functions from Fixed-Key Blockciphers. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 433–450. Springer, Heidelberg (2008)
33. Shrimpton, T., Stam, M.: Building a Collision-Resistant Compression Function from Non-compressing Primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008)
34. Simon, D.R.: Findings Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
35. Wang, L.: Personal correspondence (2009)
36. Yoneyama, K., Miyagawa, S., Ohta, K.: Leaky Random Oracle (Extended Abstract). In: Provable Security – ProvSec 2008. LNCS, vol. 5324, pp. 226–240 (2008)

On the Security of Padding-Based Encryption Schemes
— or —
Why We Cannot Prove OAEP Secure in the Standard Model

Eike Kiltz* and Krzysztof Pietrzak

Cryptology & Information Security Group
CWI Amsterdam, The Netherlands
{pietrzak,kiltz}@cwi.nl

Abstract. We investigate the security of "padding-based" encryption schemes in the standard model. This class contains all public-key encryption schemes where the encryption algorithm first applies some invertible public transformation to the message (the "padding"), followed by a trapdoor permutation. In particular, this class contains OAEP and its variants.

Our main result is a black-box impossibility result showing that one cannot prove any such padding-based scheme chosen-ciphertext secure even assuming the existence of ideal trapdoor permutations. The latter is a strong ideal abstraction of trapdoor permutations which inherits all security properties of uniform random permutations.

Keywords: Padding-based encryption, OAEP, black-box, ideal trapdoor permutations.

1 Introduction

1.1 Padding Schemes for Encryption

Optimal Asymmetric Encryption Padding (OAEP) is one of the most known and widely deployed asymmetric encryption schemes. It was designed by Bellare and Rogaway [3] as a scheme based on a trapdoor permutation (TDP). 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. After the proposal of OAEP, several variants were proposed, such as Shoup's OAEP+ [41], Boneh's Simplified OAEP [8] (SAEP), and many others (e.g., [1,8,12,13,16,30,29,34,36,37]). All the aforementioned schemes can be classified as "padding-based encryption

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* Supported by the research program Sentinels Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 389–406, 2009.

schemes": the encryption algorithm first applies a public injective transformation π to message m and randomness r, and then a trapdoor permutation f to the result, i.e., $\mathsf{Enc}(m;r) = f(\pi(m,r))$. Decryption inverts the trapdoor permutation and then applies an inverse transformation $\hat{\pi}$ to reconstruct the message (or output a special rejection symbol), i.e., $\mathsf{Dec}(c) = \hat{\pi}(f^{-1}(c))$. The two public transformations $\Pi = (\pi, \hat{\pi})$ (with the consistency requirement $\hat{\pi}(\pi(m,r)) = m$, for all m, r) are called a padding scheme. For example, in the case of OAEP the padding Π consists of a two round Feistel network, involving two hash functions.

Despite their practical importance, the only known security results for all known padding-based encryption schemes are in the random oracle model [2], where one assumes the existence of "ideal hash functions." For example, in the random oracle model, OAEP+ is secure against chosen-ciphertext attack (IND-CCA secure [38]) under the assumption that the TDP is one-way [41]; OAEP is IND-CCA secure under the (stronger) assumption that the TDP is partial-domain one-way [18]. However, such proofs merely provide heuristic evidence that breaking the scheme may be hard in reality, where the random oracles must be instantiated with some efficient hash functions. Moreover, a growing number of papers raised theoretical concerns regarding the soundness of the random oracle model. (See, e.g., [10,24].) This leaves the question whether or not padding-based encryption schemes can be securely instantiated in the standard model based on reasonable assumptions to the underlying trapdoor permutation. Or, the same question applied to OAEP: can we securely instantiate OAEP's random oracles assuming the TDP fulfils some strong security assumption beyond (partial-domain) one-wayness?

Ideal Trapdoor Permutations. We consider keyed trapdoor permutations TDP (which formally consist of key-generation, evaluation, and inversion algorithms). Extending Dodis et. al. [17], we propose the notion of an *ideal trapdoor permutation*. Informally, an ideal trapdoor permutation is a TDP that inherits *all* security properties of a uniformly random permutation. More concretely, TDP is an ideal trapdoor permutation if it satisfies all game-based security properties which are satisfied by random permutations. We stress that our basic definition only covers games where the challenger is not given the trapdoor to invert the TDP.[1]

Ideal TDPs are very powerful cryptographic primitives: by definition their security properties include, for example, (exponentially-hard) one-wayness, partial-domain one-wayness, claw-freeness, pseudo-randomness, and many other notions; from an ideal TDP we can build (in a black-box way) most cryptographic primitives, including collision-resistant hashing, pseudorandom generators (PRGs) and functions (PRFs), digital signatures, perfectly one-way hash functions (POWHFs), and, in particular, IND-CCA secure public-key encryption.

Let us remark that conceptually ideal TDPs are quite different from other idealized models like the random oracle model or the ideal cipher model. Informally,

[1] Note that allowing the challenger to access the trapdoor to invert the TDP would make it possible to model the IND-CCA security experiment itself as such a game, and thus security of schemes like OAEP would trivially follow from the fact that they are secure in the random oracle model.

the latter two models refer to particular objects (e.g., the random oracle model assumes that all parties have access to an oracle realizing a uniformly random function), whereas an ideal TDP is defined by its security properties. Although we also realize an ideal TDP via an oracle (basically, an ideal cipher with some extra functionalities modelling the trapdoors), we can add other inefficient oracles (in our case an oracle breaking all padding based encryption schemes) and –if this oracle does not affect the security properties of an ideal TDP– still are in the "ideal TDP" model.

Black-box reductions. Usually, when one constructs a cryptographic primitive A (e.g., a PRG) out of another cryptographic primitive B (e.g., a OWF), A uses B as a subroutine, independent of the particular implementation of B. The security proof for A constructs an adversary for B using any adversary for A as a subroutine. This is known as a "black-box reduction from primitive A to B" [27,26]. Black-box reductions play an important role in cryptography since almost all reductions are black-box. A black-box separation result means that there exists no black-box reduction from A to B. The common interpretation of these results is that there are inherent limitations in building primitive A from B, and that these impossibility results can be overcome only by explicitly using the code of primitive B in the construction. Although there are quite a few cryptographic constructions which are not black box —in our context, the most notable is the Naor-Yung paradigm to construct IND-CCA secure cryptosystem from any enhanced trapdoor permutation [33,31]— such constructions are usually prohibitively inefficient (e.g., the Naor-Yung paradigm relies on non-interactive zero knowledge proofs), and thus mostly of theoretical interest.

1.2 Results

Main Result. Our main result is a negative one. We show that there is no instantiation of a padding-based encryption scheme that allows a *black-box reduction* from ideal trapdoor permutations to its IND-CCA security. That is, we consider all possible padding-based encryption schemes where the padding scheme $\Pi = \Pi^{\mathsf{TDP}}$ is an oracle circuit having arbitrary access to the underlying trapdoor permutation TDP (and Π may even arbitrarily depend on the trapdoor-key of the final permutation). None of these constructions can be proved IND-CCA secure based on the security properties of the ideal trapdoor permutation TDP, using a black-box reduction.[2] As already discussed before, this hints some inherent limitations in the design concept of padding-based encryption schemes in general, and of OAEP, in particular.

Let us stress that *efficient* ideal trapdoor permutations do not exist, thus showing a black-box construction of some primitive from ideal TDPs would have limited practical relevance. (Somewhat similar to, say, a proof in the random oracle model.) But keep in mind, that we prove an impossibility result, and showing that no padding-based encryption scheme can be proved secure from ideal TDPs, immediately implies that no such scheme can be proven secure assuming a TDP which

[2] Since we require that $\hat{\pi}$ from $\Pi = (\pi, \hat{\pi})$ is publicly invertible, we avoid that the padding itself is already an IND-CCA secure encryption scheme.

has some subset of the security properties of ideal TDPs (where the subset is such that it potentially could be satisfied by some efficient construction).

Technical overview. To obtain our separation result, we describe two oracles, T and B, such that T implements an ideal trapdoor permutation. Furthermore, given access to oracle B, an adversary can (trivially) break the IND-CCA security of any padding-based encryption scheme. Yet, B does not help an adversary to break the security properties of the ideal trapdoor permutation. Now our main result can be derived using the "two-oracle separation technique" by Hsiao and Reyzin [26]. (Informally, since a black-box security proof would also have to be valid relative to the two oracles T and B, such a proof cannot exists.)

Impact on OAEP and Related Schemes. One direct application of our general theorem is that OAEP is unprovable, even assuming that (i) the TDP (used for the final encryption) has any security property satisfied by ideal TDPs; (ii) one makes any computational assumption on the hash functions in the Feistel network, such that hash functions with this security properties can be constructed from ideal TDPs.

This in particular rules out the instantiability of OAEP from most cryptographic primitives like (partial-domain) one-way trapdoor permutations, collision-resistant hash functions, PRGs, PRFs, POWHFs, and more. (Since all these primitives are black-box implied by ideal TDPs.) Our interpretation of this result is that, in order to prove IND-CCA security of OAEP in the standard model, one has to rely on a security property of the underlying TDP that is not fulfilled by ideal TDPs[3]; or, one has to make use of special non black-box properties of the employed trapdoor permutation (e.g., if one uses the RSA permutation one may try to use the fact that it is homomorphic and random self-reducible).

We stress that, even though our results indicate certain limitations in the design of OAEP and related schemes, we do not show that they are insecure, nor that one of the security claims from [41,18] are incorrect. In particular, OAEP's *random oracle model* security proof [18] can still be viewed as a valid argument supporting OAEP's security in practice. In particular, there is no "generic attack" on OAEP which treats the hash functions like random oracles.

Extensions. For the sake of simplicity we first prove our basic impossibility result as outlined above. In the full version of this paper we will then discuss how this result can be strengthened in several ways, in particular:

- We observe that our impossibility proof does not actually need the full power of IND-CCA attacks, and thus we already exclude the possibility of proving security under a significantly weaker notion. Although this notion is somewhat artificial, it contains (and thus we rule out) natural notions such as security in the sense of IND-CCA1 (lunchtime) and NM-CPA (non-malleability).
- Following [17], we extend our results to *ideal trapdoor permutations with bounded inversions*. The latter is like an ideal TDP, but in the game defining

[3] This could either be security properties where the challenger has access to the trapdoor and thus can invert the permutation or properties *not* satisfied by random permutations.

the security property, the challenger is additionally allowed to invert f on an a-priori bounded number of points. We remark that ideal TDPs with bounded inversion black-box imply the important cryptographic primitive of verifiable random functions [32] (VRFs).

– A permutation $f(\cdot)$ is homomorphic, if from $f(x), f(y)$ one can efficiently compute $f(x \circ y)$ (for some group operation $\circ$). The homomorphic property (of, e.g., RSA) has proved very useful and was exploited in numerous security proofs. As ideal TDPs are *not* homomorphic, our main result does not rule out the possibility of basing the security of some padding-based encryption scheme on the homomorphic property of the underlying TDP. Unfortunately, we show that this is not the case, as our impossibility result still holds if we add some additional oracle which imposes a homomorphic structure on the ideal TDP.

1.3 Related Work

Black-box Separations. After Impagliazzo and Rudich [27] showed that there are no black-box constructions of key-agreement protocols from one-way permutations, substantial additional work in this line followed (see, for example [19,20,22,28,42], and many more). To obtain our separation result, we use the direct "two-oracle separation technique" by Hsiao and Reyzin [26]. Most relevant to our result is the work of Dodis et. al. [17], who consider the security of full-domain hash signatures in the standard model. They showed that there is no instantiation of full-domain hash signatures that can be proven secure based on black-box access to (in our language) ideal trapdoor permutations. Also related is the separation result by Gertner et. al. [21] on the black-box impossibility of constructing IND-CCA from IND-CPA secure public-key encryption without using "re-encryption" (i.e., where decryption of the IND-CCA secure scheme is not allowed to use encryption of the IND-CPA secure scheme).

(In)security of OAEP. Due to its practical importance, a growing number of papers consider the security properties of OAEP. Revisiting the earlier security claims by Bellare and Rogaway [3], Shoup [41] showed that OAEP is black-box unprovable solely based on the one-wayness of the underlying TDP, even in the random oracle model. Later this result got complemented in [18] by showing that, in the random oracle model, one needs to assume the stronger security assumption of *partial-domain* one-wayness to prove OAEP secure.

In a series of two papers [6,7], Boldyreva and Fischlin considered the question of instantiating the random oracles in OAEP (and other scenarios) by specific candidates of standard-model hash functions, such as POWHFs and VRFs. In particular, they showed that POWHFs or VRFs cannot generically instantiate the random oracles in OAEP, no matter which TDP is used [6]. Although it follows immediately from our generic impossibility result that one cannot *prove the security* of OAEP (or any other padding-based scheme) assuming the hash functions are instantiated with such primitives, the result of [6] (for the special case of OAEP) is stronger as they show concrete instantiations which actually

make OAEP insecure. On the positive side, [7] show that if the hash functions in OAEP are instantiated using non-malleable pseudorandom generators, then the resulting OAEP scheme is proved non-malleable. However, their security definition of non-malleability is actually weaker than what is commonly called NM-CPA security [4] which is the reason why their positive instantiation result does not contradict our separation results.[4]

Brown [9] showed that RSA-OAEP cannot be proven CCA secure under a certain class of security reductions denoted as "key-preserving" black-box reductions, i.e., reductions that are restricted to make oracle calls to the CCA adversary with respect to the *same RSA instance* that they are given as challenge. Similar results (for the class of "single-key factoring-based encryption schemes") were independently obtained by Paillier and Villar [35]. Our impossibility results seem more general since we can exclude any black-box reduction (and not only key-preserving ones) from ideal trapdoor permutations (and not only one-wayness). Furthermore, the results from [9,35] do not allow the scheme's public key to contain any additional information beyond the RSA/factoring instance. In particular, their results do not exclude the possibility to securely instantiate OAEP from standard *keyed* hash functions, such as POWHFs and VRFs.

2 Preliminaries

2.1 Notation

If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. If x is a string, then $[x]_\ell$ denote the ℓ left-most bits of x. If S is a set then $s \leftarrow_R S$ denotes the operation of picking an element s of S uniformly at random. We write $\mathsf{A}(x, y, \ldots)$ to indicate that A is an algorithm (i.e., a Turing Machine) with inputs $x, y, \ldots$ and by $z \leftarrow_R \mathsf{A}(x, y, \ldots)$ we denote the operation of running A with inputs $(x, y, \ldots)$ and letting z be the output. We write $\mathsf{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ to indicate that A is an algorithm with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$. With PT we denote polynomial time and with PPT we denote probabilistic polynomial time.

2.2 Public-Key Encryption

A *public key encryption* scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\{0,1\}^\mu$ (where $\mu = \mu(k)$ is some polynomial in k) consists of three PT algorithms, of which the first two, Kg and Enc, are probabilistic and the last one, Dec, is deterministic. Public/secret keys for security parameter $k \in \mathbb{N}$ are generated using

[4] The non-malleability definitions of [7] only consider relations over one ciphertext and not over polynomial-size ciphertext vectors. Using the characterization of [4], one can actually prove that this is an even weaker notion than IND-CPA security where the adversary is allowed to additionally make one single decryption query (also called 1-bounded IND-CCA-security [14]). Our results show that full NM-CPA security [4] is not black-box achievable. We remark that the security notion of [7] is still meaningful since it prevents Bleichenbacher's attack on PKCS [5].

$(pk, sk) \leftarrow_R \mathsf{Kg}(1^k)$. Given such a key pair, a message $m \in \{0,1\}^\mu$ is encrypted by $c \leftarrow_R \mathsf{Enc}(pk, m)$; a ciphertext is decrypted by $m \leftarrow \mathsf{Dec}(sk, c)$, where possibly Dec outputs a special reject symbol $\perp$ to denote an invalid ciphertext. For correctness, we require that for all $k \in \mathbb{N}$, all messages $m \in \{0,1\}^\mu$, it must hold that $\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1$, where the probability is taken over the above randomized algorithms and $(pk, sk) \leftarrow_R \mathsf{Kg}(1^k)$. Sometimes we also associate a randomness space $\{0,1\}^\rho$ to PKE (where $\rho = \rho(k)$ is some polynomial in k), to denote that the randomness used in Enc is picked uniformly from $\{0,1\}^\rho$.

We recall the standard security notion of *chosen ciphertext security* [38] of a PKE scheme which is defined through the following advantage function of an adversary A.

$$\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{PKE}}(\mathsf{A}) = \left| \Pr\left[b = b' : \begin{array}{l} (pk, sk) \leftarrow_R \mathsf{Kg}(1^k) \\ (m_0, m_1, state) \leftarrow_R \mathsf{A}^{\mathcal{O}(sk,\cdot)}(pk) \\ b \leftarrow_R \{0,1\}\,;\ c^* \leftarrow_R \mathsf{Enc}(pk, m_b) \\ b' \leftarrow_R \mathsf{A}^{\mathcal{O}(sk,\cdot)}(c^*, state) \end{array} \right] - \frac{1}{2} \right| ,$$

where $\mathcal{O}(sk, c) = \mathsf{Dec}(sk, c)$, in the second phase ("guess phase"), A is not allowed to query $\mathcal{O}(sk, \cdot)$ for the challenge ciphertext c^*, and we require that m_0 and m_1 are of the same length. *state* is some arbitrary state information. PKE scheme PKE is said to be chosen ciphertext secure ($\mathsf{IND\text{-}CCA}$ secure) if the advantage function $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{PKE}}(\mathsf{A})$ is a negligible function in k for all PPT adversaries A.

3 Ideal Trapdoor Permutations

In this section we introduce the notion of an ideal trapdoor permutation. Intuitively, this is a trapdoor permutation that inherits all security properties of a *uniformly random* permutation. This includes (partial-domain) one-wayness, claw-freeness, and other non-standard notions.

3.1 Trapdoor Permutations

Definition 1. *A triple of PPT algorithms* $(\mathsf{Tdg}, \mathsf{F}, \mathsf{F}^{-1})$ *implements a trapdoor permutation if* Tdg *is probabilistic and on input* 1^k *generates an evaluation/trapdoor key-pair* $(ek, td) \leftarrow_R \mathsf{Tdg}(1^k)$, $\mathsf{F}(ek, \cdot)$ *implements a permutation* $f_{ek}(\cdot)$ *over* $\{0,1\}^k$ *and* $\mathsf{F}^{-1}(td, \cdot)$ *implements its inverse* $f^{-1}_{ek}(\cdot)$.

Note that the above definition is only functional, and does not impose any security property. The most common security property for TDPs is one-wayness, i.e., one requires that it is hard to invert the permutation on random inputs without knowing the trapdoor. We will consider a much broader class of security properties. Using the notion of δ-*hard games* (generalizing a notion previously used in [17]) we can capture any game-based cryptographic hardness experiment involving permutations.

3.2 Hard Games

A *game* is defined by a PPT algorithm G (the challenger). This G can interact with another PPT algorithm A (the adversary) by exchanging messages over a shared communication tape. Eventually, G outputs a decision bit d. We denote one execution of game G with adversary A by $d \leftarrow_R \mathbf{Exp}^{\mathsf{G}}(\mathsf{A})$ and say that A *wins game* G if $d = 1$.

A game G as above defines a δ-*hard game*, where $0 \leq \delta < 1$, if G is a PPT algorithm (in the security parameter $k \in \mathbb{N}$), and further no PPT adversary A can win the game when both, G and A, have oracle access to $t = t(k)$ uniform random permutations $\tau_1, \ldots, \tau_t$ over $\{0,1\}^k$ (here $t(\cdot)$ is implicitly defined by G) with probability significantly better than δ. Formally, we define the advantage function of an adversary A in game G as

$$\mathbf{Adv}^{\mathsf{G}}_{\mathrm{RP}}(\mathsf{A},k) = \Pr\left[d = 1 \ : \ d \leftarrow_R \mathbf{Exp}^{\mathsf{G}^{\tau_1(\cdot),\ldots,\tau_t(\cdot)}}(\mathsf{A}^{\tau_1(\cdot),\ldots,\tau_t(\cdot)}(1^k))\right] . \quad (1)$$

The RP in $\mathbf{Adv}^{\mathsf{G}}_{\mathrm{RP}}(\mathsf{A},k)$ stands for "random permutation". Let us stress once more, that in the above game we do not give G (or A) access to the inversion oracles $\tau_1^{-1}(\cdot), \ldots, \tau_t^{-1}(\cdot)$.

Definition 2. *Game* G *is* δ*-hard for some* $0 \leq \delta \leq 1$*, if for all PPT adversaries* A*,* $\mathbf{Adv}^{\mathsf{G}}_{\mathrm{RP}}(\mathsf{A},k) - \delta$ *is negligible in* k*. The hardness of a game* G*, denoted* $\delta(\mathsf{G})$*, is the smallest* δ *such that* G *is* δ*-hard.*

Table 1 shows examples of hard games with the corresponding upper bound on the advantage function $\mathbf{Adv}^{\mathsf{G}}_{\mathrm{RP}}(\mathsf{A})$. Typical values for $\delta(\mathsf{G})$ are 0 and 1/2, the latter comes up in games where the adversary just has to distinguish two cases, and thus can trivially win with probability 1/2 by a random guess. The notion of δ-hard games generalizes the hard-games used in [17], which only covered the case $\delta = 0$, but the results (and proofs) from [17] can easily be adapted to cover general δ-hard games.

3.3 Ideal Trapdoor Permutations

A trapdoor permutation $\mathsf{TDP} = (\mathsf{Tdg}, \mathsf{F}, \mathsf{F}^{-1})$ is secure for hard game G if Definition 2 is satisfied even if the random permutations used to define (1) are replaced with instantiations of TDP. Let

$$\mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}}(\mathsf{A},k) := \Pr\left[d = 1 : \begin{array}{l} \text{for } i = 1,\ldots,t : \quad (ek_i, td_i) \leftarrow_R \mathsf{Tdg}(1^k)\,; \\ d \leftarrow_R \mathbf{Exp}^{\mathsf{G}^{\mathsf{F}(ek_1,\cdot),\ldots,\mathsf{F}(ek_t,\cdot)}}(\mathsf{A}(ek_1,\ldots,ek_t)) \end{array}\right]$$

Definition 3. *A trapdoor permutation* TDP *is* secure for game G *if for all PPT adversaries* A*,* $\mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}}(\mathsf{A},k) - \delta(\mathsf{G})$ *is negligible in* k*. Furthermore,* TDP *is an* ideal trapdoor permutation *if it is secure for all games.*

Table 1. Examples of hard games. In the last column, q is an upper bound on the number of queries the adversary A makes to each of its oracles $\tau_1(\cdot), \ldots, \tau_t(\cdot)$. Note that if $q = q(k)$ is polynomially bounded, then in all cases $\mathbf{Adv}^{\mathsf{G}}_{\mathrm{RP}}(\mathsf{A}, k) - \delta(\mathsf{G})$ is negligible (even exponentially small), thus the games are δ-hard (with $\delta = 0$ or $\delta = 1/2$ as indicated in the table).

Game	Description of game $\mathsf{G}^{\tau_1,\ldots,\tau_t}$			Advantage
	Experiment	Winning cond.	$\delta(\mathsf{G})$	$\mathbf{Adv}^{\mathsf{G}}_{\mathrm{RP}}(\mathsf{A}, k)$
One-wayness (OW)	$x \leftarrow_R \{0,1\}^k$; $y \leftarrow \tau_1(x)$; $x' \leftarrow_R \mathsf{A}^{\tau_1}(y)$	$x' = x$	0	$\leq (q+1)/2^k$
Partial-domain OW	$x \leftarrow_R \{0,1\}^k$; $y \leftarrow \tau_1(x)$; $x' \leftarrow_R \mathsf{A}^{\tau_1}(y)$	$x' = [x]_\ell$	0	$\leq \frac{q}{2^k} + \frac{2^{k-\ell}}{2^k - q}$
Claw-freeness	$(x_1, x_2) \leftarrow_R \mathsf{A}^{\tau_1,\tau_2}(1^k)$	$\tau_1(x_1) = \tau_2(x_2)$	0	$\leq q^2/2^k$
Pseudorandomness	$x \leftarrow_R \{0,1\}^k$; $y_1 \leftarrow \tau_1(x)$; $b \leftarrow_R \{0,1\}$ $y_{2,0} \leftarrow_R \{0,1\}^k$; $y_{2,1} \leftarrow \tau_2(x)$ $b' \leftarrow_R \mathsf{A}^{\tau_1,\tau_2}(y_1, y_{2,b})$	$b' = b$	$\frac{1}{2}$	$\leq \frac{1}{2} + \frac{q}{2^k}$
$t(k)$-correlated input OW	$x \leftarrow_R \{0,1\}^k$; for $i = 1, \ldots, t : y_i \leftarrow \tau_i(x)$ $x' \leftarrow_R \mathsf{A}^{\tau_1,\ldots,\tau_t}(y_1, \ldots, y_t)$	$x' = x$	0	$\leq \frac{1+t\cdot q}{2^k}$

3.4 On the Power of Ideal Trapdoor Permutations

Ideal trapdoor permutations are a quite powerful primitive as most of the known cryptographic primitives can be efficiently instantiated in a *black-box way* from ideal trapdoor permutations. Examples include: collision-resistant hashing (from claw-freeness [15]); pseudorandom generators and functions (from one-wayness, using the Goldreich-Levin predicate [23]); perfectly one-way probabilistic hash functions (from one-wayness [11]); IND-CCA secure public-key encryption (from $t(k)$-correlated input one-wayness [39] for $t(k) = 2k+1$); bit commitment (from one-wayness), digital signatures, oblivious transfer [25], trapdoor commitments, and many more.

On the other hand, it is easy to see (see [17] for a proof) that ideal trapdoor permutations do not exists. However, keep in mind that we are aiming for an impossibility result: we rule out the existence of padding-based encryption schemes whose security can be black-box reduced to ideal TDPs. This will immediately also rule out this possibility for any TDP which are only hard for a "realistic" subset of all hard games.

4 Padding Schemes for Encryption

In this section we introduce the notion of padding-schemes and padding-based encryption schemes. Many efficient and widely employed encryption schemes, in particular OAEP and its variants, are padding-based.

4.1 Definitions

Let k, μ, ρ be three integers such that $\mu + \rho \leq k$. A *padding scheme* Π consists of two mappings $\pi : \{0,1\}^\mu \times \{0,1\}^\rho \to \{0,1\}^k$ and $\hat{\pi} : \{0,1\}^k \to \{0,1\}^\mu \cup \{\perp\}$ such that π is injective and the following consistency requirement is fulfilled:

$$\forall m \in \{0,1\}^\mu, r \in \{0,1\}^\rho \ : \quad \hat{\pi}(\pi(m \,\|\, r)) = m \ .$$

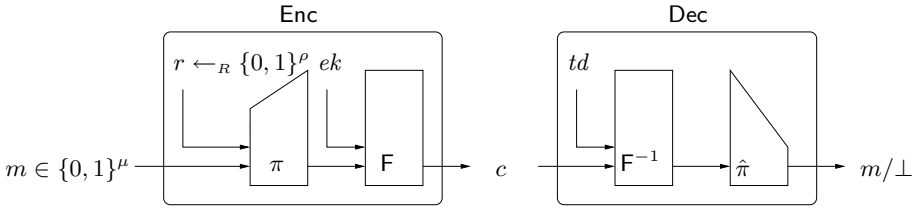


Fig. 1. Padding based encryption scheme from a trapdoor permutation $\mathsf{TDP} = (\mathsf{Tdg}, \mathsf{F}, \mathsf{F}^{-1})$

A pair of oracle circuits $\Pi = (\pi, \hat{\pi})$ is a *TDP-based padding scheme*, if $\pi^{\mathsf{TDP}}, \hat{\pi}^{\mathsf{TDP}}$ is a padding scheme for any trapdoor permutation $\mathsf{TDP} = (\mathsf{Tdg}, \mathsf{F}, \mathsf{F}^{-1})$.

Definition 4. *Let $\mu, \rho : \mathbb{N} \to \mathbb{N}$ be functions (defining a message and randomness space, respectively) where $\mu(k) + \rho(k) \leq k$ for all k. We call a triple of efficient oracle algorithms $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ a* padding-based encryption scheme *with message space μ and randomness space ρ, if for any trapdoor permutation $\mathsf{TDP} = (\mathsf{Tdg}, \mathsf{F}, \mathsf{F}^{-1})$:*

- *(Key Generation) Given the security parameter k, $\mathsf{Kg}^{\mathsf{TDP}}(k)$ returns a public key $pk = (ek, \Pi)$ and a secret key $sk = (td, \hat{\pi})$, where ek and td are computed by running $(ek, td) \leftarrow_R \mathsf{Tdg}(1^k)$ and $\Pi = (\pi, \hat{\pi})$ is a TDP based padding scheme.*
- *(Encryption) Given the public-key pk and a message $m \in \{0,1\}^\mu$, $\mathsf{Enc}^{\mathsf{TDP}}(pk, m)$ returns a ciphertext computed as $c = f_{ek}(\pi^{\mathsf{TDP}}(m \,\|\, r)) \in \{0,1\}^k$, for $r \leftarrow_R \{0,1\}^\rho$.*
- *(Decryption) Given the secret-key sk and a ciphertext $c \in \{0,1\}^k$, $\mathsf{Dec}^{\mathsf{TDP}}(sk, c)$ returns $m = \hat{\pi}^{\mathsf{TDP}}(f_{ek}^{-1}(c)) \in \{0,1\}^\mu \cup \{\perp\}$.*[5]

See Fig. 1 for a graphical illustration of Definition 4. Note that it is not further specified how $\mathsf{Dec}^{\mathsf{TDP}}(sk, \cdot)$ behaves on invalid ciphertexts, i.e., on a c which is not in the range of $\mathsf{Enc}^{\mathsf{TDP}}(pk, \cdot)$. $\mathsf{Dec}^{\mathsf{TDP}}(sk, \cdot)$ may return the special reject symbol $\perp$, or output any message, depending on the definition of $\hat{\pi}$.

Also note that we include $\hat{\pi}$ as part of the public-key, even though $\hat{\pi}$ is not required in the encryption algorithm at all. This apparently minor detail is important as we will explain later in Section 5. Basically, by including $\hat{\pi}$ in pk, we make sure that the padding scheme $\Pi = (\pi, \hat{\pi})$ *itself* is not already an IND-CCA secure encryption scheme.

4.2 Examples

Our definition of padding-based encryption schemes is quite broad and contains many known encryption schemes, including OAEP and its variants. Fig. 2 contains the description of π for the underlying padding scheme Π, for some

[5] Recall that we use $f_{ek}(\cdot) := \mathsf{F}(ek, \cdot)$ and $f_{ek}^{-1}(\cdot) := \mathsf{F}^{-1}(td, \cdot)$, where td is the trapdoor corresponding to ek.

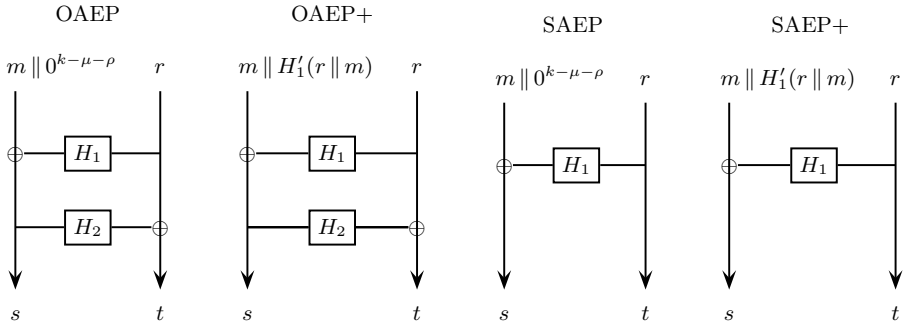


Fig. 2. Examples of the mapping π for important padding schemes. Here $H_1 : \{0,1\}^{\rho} \to \{0,1\}^{k-\rho}$, $H_1' : \{0,1\}^{k-\rho} \to \{0,1\}^{k-\mu-\rho}$, $H_2 : \{0,1\}^{k-\rho} \to \{0,1\}^{\rho}$ are hash functions (whose circuit description are contained in the description of π).

important schemes. The corresponding inverse $\hat{\pi}$ can be easily derived from Π's consistency property. For example, in OAEP, $\hat{\pi}(s \,\|\, t)$ is defined as $\perp$ or m, depending whether $w = 0^{k-\mu-\rho}$, or not, where $m \,\|\, w = s \oplus H_1(H_2(s) \oplus t))$. Other examples of padding-based encryption schemes not contained in Fig. 2 include OAEP++ [29], PSS-E [13], PSP2 S-Pad [16], full-domain permutation encryption [36], 3-round OAEP [36,37], 4-round OAEP [1], and the schemes from [12,30,34].

5 Uninstantiability from any Ideal Trapdoor Permutation

The following main theorem states that there does not exist a padding-based encryption scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ such that any adversary who breaks the IND-CCA security of $\mathsf{PKE}^{\mathsf{TDP}}$ can be used (in a black-box way) to break the security of TDP as an ideal TDP.

Theorem 5. *There is no black-box reduction from an ideal trapdoor permutation to a chosen-ciphertext (*IND-CCA*) secure padding-based encryption scheme.*

Proof. Fix a padding-based encryption scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mu(k)$ and randomness space $\rho(k)$. If the message plus the randomness space is too small, i.e., $\mu(k)+\rho(k) \in O(\log k)$ or equivalently $2^{\mu(k)+\rho(k)} \in poly(k)$, then $\mathsf{PKE}^{\mathsf{TDP}}$ is trivially insecure to matter what TDP we use, as an adversary in the IND-CCA experiment can trivially decrypt the challenge ciphertext by doing an exhaustive search over all possible plaintext and randomness pairs $(m, r) \in \{0,1\}^{\mu(k)+\rho(k)}$, without even using the decryption oracle. Such an adversary runs in $2^{\mu(k)+\rho(k)} \in poly(k)$ time units and has success probability 1 in breaking the IND-CCA security of PKE.

So from now on we can assume (w.l.o.g.) $2^{\mu(k)+\rho(k)} \notin poly(k)$. Following [26, Proposition 1], as to rule out black-box reductions, it is enough to prove that there exist two oracles T and B such that the following holds:

1. T can be used to implement a trapdoor permutation, i.e., there exists a triple of oracle PPT algorithms $\mathsf{TDP} = (\mathsf{Tdg}, \mathsf{F}, \mathsf{F}^{-1})$ such that $\mathsf{TDP}^{\mathsf{T}}$ implements a trapdoor permutation (in the sense of Definition 1).
2. Relative to the oracles T and B, $\mathsf{TDP}^{\mathsf{T}}$ is an *ideal trapdoor permutation*. That is, $\mathsf{TDP}^{\mathsf{T}}$ is an ideal trapdoor permutation as in Definition 3 even if the adversary is given access to the oracles T and B.
3. For any padding-based encryption schemes PKE: relative to the oracles T and B, $\mathsf{PKE}^{\mathsf{TDP}^{\mathsf{T}}}$ is not $\mathsf{IND\text{-}CCA}$ secure.

We first define the oracle T and show that it satisfies point 1 (Lemma 6) and a relaxed version of point 2 where we do not consider the breaking oracle B (Lemma 7). We then define the breaking oracle B, and prove that points 2 and 3 hold (Lemma 10 and Lemma 8, respectively).

Definition of T. (Oracle used to implement the trapdoor permutation.) Let $\mathcal{P}_k$ denote the set of all permutations over $\{0,1\}^k$. For any $k \in \mathbb{N}$, choose $2^k + 1$ permutations $f_{k,0}, \ldots, f_{k,2^k-1}$ and g_k from $\mathcal{P}_k$ uniformly at random. $\mathsf{T} = (\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3)$ is defined as follows.

- $\mathsf{T}_1(td)$ returns $g_k(td)$, where $k = |td|$. (Convert trapdoor into public key)
- $\mathsf{T}_2(ek, x)$ with $|ek| = |x|$ returns $f_{k,ek}(x)$, where $k = |x|$. (Evaluation)
- $\mathsf{T}_3(td, y)$ with $|td| = |y|$ returns $f^{-1}_{k,g_k(td)}(y)$, where $k = |y|$. (Inversion)

Lemma 6. *There is a PPT* TDP *such that* $\mathsf{TDP}^{\mathsf{T}}$ *implements a trapdoor permutation.*

Proof. We implement a trapdoor permutation $\mathsf{TDP}^{\mathsf{T}} = (\mathsf{Tdg}, \mathsf{F}, \mathsf{F}^{-1})$ as follows.

- $\mathsf{Tdg}(1^k)$ first picks a random trapdoor $td \leftarrow_R \{0,1\}^k$, then computes the corresponding key $ek = \mathsf{T}_1(td)$, and outputs (ek, td).
- $\mathsf{F}(ek, x)$ returns $\mathsf{T}_2(ek, x)$.
- $\mathsf{F}^{-1}(td, y)$ returns $\mathsf{T}_3(td, y)$.

According to Definition 1 this implements a trapdoor permutation. ▲

Lemma 7. $\mathsf{TDP}^{\mathsf{T}}$ *is an ideal trapdoor-permutation with probability* 1 *(over the choice of* T*).*

Proof. Consider any δ-hard game G (assume for now that $t = 1$, i.e., the game involves just one permutation). Recall that the advantage of an adversary A in winning the game is

$$p_{real} := \Pr\left[d = 1 : (ek, td) \leftarrow_R \mathsf{Tdg}(1^k)\ ;\ d \leftarrow_R \mathbf{Exp}^{\mathsf{G}^{\mathsf{F}(ek,\cdot)}}(\mathsf{A}^{\mathsf{TDP}^{\mathsf{T}}}(ek))\right] \quad (2)$$

Now let T_{ek} denote T, where $f_{k,ek}(\cdot)$ is replaced with a "fresh" random permutation $\tau(\cdot)$. Let

$$p_{rand} := \Pr\left[d = 1 : \begin{array}{l}(ek, td) \leftarrow_R \mathsf{Tdg}(1^k)\ ;\ \tau(\cdot) \leftarrow_R \mathcal{P}_k\ ; \\ d \leftarrow_R \mathbf{Exp}^{\mathsf{G}^{\tau(\cdot)}}(\mathsf{A}^{\mathsf{TDP}^{\mathsf{T}_{ek}}}(ek))\end{array}\right] \quad (3)$$

By definition of a δ-hard game we have $p_{rand} - \delta \leq negl(k)$. Below we will show that $|p_{real} - p_{rand}| = negl(k)$, and thus $p_{real} - \delta \leq negl(k)$. As this holds for any δ-hard game, $\mathsf{TDP}^{\mathsf{T}}$ is an ideal TDP.

To show that $|p_{real} - p_{rand}| = negl(k)$, we will argue that A cannot distinguish the two games considered in (2) and (3) with non-negligible probability. First, as a random permutation is one way almost certainly (see [19] for a proof), an efficient adversary will find the trapdoor td given $ek = \mathsf{T}_1(td)$ only with exponentially small probability. (In particular, A will almost certainly never query the inversion oracle F^{-1} with trapdoor td.) Second, one can show (we omit the proof) that a permutation chosen at random from a set of exponentially many permutations (in particular, $f_{k,ek}$ for $ek \in \{0,1\}^k$) can be distinguished from a randomly chosen permutation by a polynomial size circuit only with an exponentially small advantage. This two points imply that the distribution of d in experiments (2) and (3) have exponentially small statistical distance, and thus $|p_{real} - p_{rand}|$ is exponentially small. ▲

Definition of B. (Oracle used to break the encryption scheme.) The oracle B is defined below. It takes two types of inputs. On input the description of a padding-based encryption scheme, B outputs a vector of challenge ciphertexts. On input a padding-based encryption scheme with a vector of plaintexts, B checks if those plaintexts correspond to the ciphertexts it would ask on a query of the first type. If this is the case, B outputs the trapdoor of the encryption scheme.

It is quite obvious how the oracle breaks the security of any padding based encryption scheme (with message space $\mu(\cdot)$ and randomness space $\rho(\cdot)$). It is less obvious that this oracle will not be of much use in winning any hard game. In order to prove this we will show that basically the only possibility of making a query of the second type to B containing the correct plaintexts (and thus receiving the trapdoor), is by already knowing the trapdoor. Note that the chosen-ciphertext security game itself cannot be formulated as a hard game since the challenger has to know the trapdoor to answer decryption queries. It is exactly this additional power of the CCA challenger (as compared to the challenger in a hard game) that allows an adversary interacting with the CCA challenger to exploit the answers of the breaking oracle.

We now formally define B. (Like all other oracles we consider, this oracle is stateless, and thus will return the same output when queried twice on an identical input.)

1. B, on an input of the form (k, ek, Π), where $k \in \mathbb{N}$, $ek \in \{0,1\}^k$, and $\Pi = \{\pi, \hat{\pi}\}$ (where $\pi : \{0,1\}^{\mu(k)+\rho(k)} \to \{0,1\}^k, \hat{\pi} : \{0,1\}^k \to \{0,1\}^{\mu(k)}$ is a TDP-based padding scheme), outputs a vector of challenge ciphertexts $[c_1, \ldots, c_{4k}]$, computed as
$$c_i = f_{k,ek}(\pi^{\mathsf{T}}(m_i \,\|\, r_i)), \tag{4}$$
for randomly chosen $m_i \leftarrow_R \{0,1\}^{\mu(k)}$ and $r_i \leftarrow_R \{0,1\}^{\rho(k)}$. Note that if (ek, Π) is the public-key of some padding based encryption scheme, then c_i is a proper ciphertext of message m_i for this public key.
2. B, on input $(k, ek, \Pi, [m'_1, \ldots, m'_{4k}])$, checks if $[m'_1, \ldots, m'_{4k}] = [m_1, \ldots, m_{4k}]$, where the m_i are the plaintext messages chosen by B on input (k, ek, π) as

described above. If the two plaintext vectors are identical, it returns $td := g_k^{-1}(ek)$, the trapdoor of the trapdoor permutation corresponding to ek from the input of B. Otherwise, it outputs $\perp$.

Lemma 8. *There is a PPT* A *such that* $\mathsf{A}^{\mathsf{T},\mathsf{B}}$ *breaks the* IND-CCA *security of* $\mathsf{PKE}^{\mathsf{TDP}^{\mathsf{T}}}$.

Proof. Adversary A in the IND-CCA experiment first obtains a public key which contains ek for the trapdoor permutation and the padding scheme Π. Next, it queries B on input (k, ek, π) to obtain the vector of challenge ciphertexts $[c_1, \ldots, c_{4k}]$. Next, A asks its decryption oracle $\mathcal{O}$ for the plaintexts $m'_i = \mathcal{O}(c_i)$, for $i = 1, \ldots, 4k$. Finally, A makes the query $(k, ek, \pi, [m'_1, \ldots, m'_{4k}])$ to B and obtains the trapdoor td which can be used to decrypt the challenge ciphertext c^* (and hence distinguish). We have just shown that A is a PPT algorithm with $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{PKE}}(\mathsf{A}) = 1$. ▲

Let us stress that it is in the proof above where we exploit the fact (as mentioned in the paragraph before Section 4.2) that any ciphertext can be decrypted given the public key $pk = (ek, \Pi = (\pi, \hat{\pi}))$ and trapdoor td. In particular, this means that $\hat{\pi}$ is efficiently computable.

By the following lemma, the breaking oracle B can be efficiently simulated in some settings. In particular, this will be the case for δ-hard games, and thus —as stated in Lemma 10 below— we are able to generalize Lemma 7 to hold relative to B.

Lemma 9. *For* $i = 1, \ldots, t$ *let* $(ek_i, td_i) \leftarrow_R \mathsf{Tdg}^{\mathsf{T}}(1^k)$. *Consider any oracle circuit* C *where* $\mathsf{C}^{\mathsf{T},\mathsf{B}}(ek_1, \ldots, ek_t)$ *makes at most* $q(k)$ *oracle queries where* $q(k) \leq 2^{\mu(k)+\rho(k)}/2$. *Then there exists an efficient simulator* S *such that the output of* $\mathsf{C}^{\mathsf{T},\mathsf{B}}(ek_1, \ldots, ek_t)$ *and* $\mathsf{C}^{\mathsf{T},\mathsf{S}^{\mathsf{T}}}(ek_1, \ldots, ek_t)$ *is exponentially close (i.e., the statistical distance is* $\leq 2^{-k/2}$*). Here* S *will get to see all* $\mathsf{F}^{-1}(\cdot,\cdot)$ *(and only those) queries made by* C *to the* T *oracle.*

Before we prove this lemma, let us see how we can use it to generalize Lemma 7.

Lemma 10. $\mathsf{TDP}^{\mathsf{T}}$ *is an ideal trapdoor-permutation with probability* 1 *(over the choice of* T*)* even relative to B.

Proof. By Lemma 7 we know that $\mathsf{TDP}^{\mathsf{T}}$ is an ideal TDP relative to T only. Now consider any game G and any adversary A, and let $q(k) \in \mathit{poly}(k)$ denote the total number of oracle queries made by G and A. As we assume $2^{\mu(k)+\rho(k)} \notin \mathit{poly}(k)$ we have $q(k) \leq 2^{\mu(k)+\rho(k)}/2$ for all but finitely many k. Thus, by Lemma 9, we can simulate B efficiently, in particular

$$\mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathsf{T}}}(\mathsf{A}^{\mathsf{T},\mathsf{B}}, k) = \mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathsf{T}}}(\mathsf{A}^{\mathsf{T},\mathsf{S}^{\mathsf{T}}}, k) \pm \mathit{negl}(k)\ . \tag{5}$$

As S is efficient, we can let its computation be done by the adversary: let $\hat{\mathsf{A}}$ denote the adversary A, but which simulates the S^{T} queries itself. (Note that this is possible, as G never makes F^{-1} queries, and S needs to see only those.)

$$\mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathsf{T}}}(\mathsf{A}^{\mathsf{T},\mathsf{S}^{\mathsf{T}}}, k) = \mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathsf{T}}}(\hat{\mathsf{A}}^{\mathsf{T}}, k) \tag{6}$$

As by Lemma 7, $\mathsf{TDP}^{\mathsf{T}}$ is an ideal TDP relative to T, $\hat{\mathsf{A}}$ cannot win the δ-hard game G with advantage more than

$$\mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathsf{T}}}(\hat{\mathsf{A}}^{\mathsf{T}}, k) \leq \delta \pm negl(k) \ . \tag{7}$$

By (5)-(7), we obtain

$$\mathbf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathsf{T}}}(\mathsf{A}^{\mathsf{T},\mathsf{B}}, k) \leq \delta \pm negl(k) \ ,$$

which implies that G is a δ-hard game, even relative to oracle B. ▲

Proof (Lemma 9)

We only consider the case $t = 1$, i.e., where there is only one single key ek. The generalization to $t \geq 1$ is straight forward. The simulator S is defined as follows.

On input of a query (k, ek, Π), S samples $m_1, \ldots, m_{4k}$ and $r_1, \ldots, r_{4k}$ and outputs $[c_1, \ldots, c_{4k}]$ computed as in (4). This query, as well as the values m_i, r_i are stored. (If the query is repeated, the same answer is given). Note that the output of this query was computed exactly the same way as B would.

On input a query $(k, ek, \Pi, [m'_1, \ldots, m'_{4k}])$, S first checks if the query (k, ek, Π) was already made.

- If this is not the case, S outputs $\perp$. Note that this almost perfectly simulates B, as B would also output $\perp$ in this case, except if by chance all the m'_i correspond to the m_i that B would use on input (k, ek, Π). (The probability of this event is $2^{-\mu(k)4k}$ what we ignore.)
- If C made the query (k, ek, Π), let $[m_1, \ldots, m_{4k}]$ denote the message vector used by S to answer this query. If $[m_1, \ldots, m_{4k}] \neq [m'_1, \ldots, m'_{4k}]$ then output $\perp$. Note that this perfectly simulates B.
- Otherwise, if $[m_1, \ldots, m_{4k}] = [m'_1, \ldots, m'_{4k}]$, S checks for all F^{-1} queries (td, x) made by C, if $ek \stackrel{?}{=} \mathsf{Tdg}(td)$. If this is the case, S outputs this trapdoor td, exactly as B would. If C never used the trapdoor td corresponding to ek in a query, S outputs "fail". Note that this is the only case where the answer from S differs from what B would output.

To prove that S can almost perfectly simulate B, it remains to upper bound the probability that an efficient adversary C can make S^{T} output "fail" in the last item above.

S outputs "fail", if C makes a query (k, ek, Π) to S^{T}, then receives $4k$ ciphertexts $[c_1, \ldots, c_{4k}]$ computed as $c_i = f_{k,ek}(\pi^{\mathsf{T}}(m_i \,\|\, r_i))$ for random m_i, r_i, and then correctly computes (or guesses) all the m_i without ever inverting $f_{k,ek}$ (i.e., never using the F^{-1} oracle with the trapdoor td where $ek = \mathsf{Tdg}(td)$). To analyze the probability that S outputs "fail", consider the following three sets.

- Let $\mathcal{R} = \{\pi^{\mathsf{T}}(m \,\|\, r) \ : \ m \,\|\, r \in \{0,1\}^{\mu(k)+\rho(k)}\}$ denote the set of possible inputs on which one must evaluate $f_{k,ek}$ in order to compute a ciphertext. As π^{T} is injective, $|\mathcal{R}| = 2^{\mu(k)+\rho(k)}$.
- Let $\mathcal{Y} = \{\pi^{\mathsf{T}}(m_i \,\|\, r_i) \ : \ i = 1, \ldots, 4k\}$ denote the set of inputs to $f_{k,ek}$ one must make in order to compute the c_i's. As π^{T} is injective, $|\mathcal{Y}| = 4k$.

– Let $\mathcal{X} \subset \mathcal{R}$ denote all the set of queries that C made to $f_{k,ek}$ (before and after seeing the challenge vector $[c_1, \ldots, c_{4k}]$).

Let $miss := |\mathcal{Y} \setminus \mathcal{X}|$ denote the number of preimages of the c_i's which C did not query. As the preimages of the c_i are uniformly in $\mathcal{R}$, and $f_{k,ek}$ is a random permutation, $miss$ is a random variable, which can be sampled as follows: from a set $\mathcal{R}$ of (super-polynomial) size $2^{\mu(k)+\rho(k)}$, sample a random subset $\mathcal{X}$ of (polynomial size) $q(k)$ and a random subset $\mathcal{Y}$ of size $4k$ and let $miss$ denote the number of elements in $\mathcal{Y}$ which are not in $\mathcal{X}$. The expected value of $miss$ is $(1 - q(k)/2^{\mu(k)+\rho(k)})4k$, which, as $q(k) \leq 2^{\mu(k)+\rho(k)}/2$, is at least $4k/2 = 2k$. Applying a Hoeffding bound, we get that the probability that $miss \geq k$ (i.e., that $miss$ is not bounded away by more than k from its expectation) is at least $1 - e^{-k/2}$.

Thus, in order to get an answer $\neq \perp$ from B, C will have to guess almost certainly at least k of the m_i's, the probability of that happening is roughly[6] $2^{-\mu(k)\cdot k} \leq 2^{-k}$. ▲

This concludes the proof of Theorem 5. ■

References

1. Abe, M., Kiltz, E., Okamoto, T.: CCA-security with optimal ciphertext overhead. In: ASIACRYPT, pp. 355–371 (2008)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73 (1993)
3. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
4. Bellare, M., Sahai, A.: Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 519–536. Springer, Heidelberg (1999)
5. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
6. Boldyreva, A., Fischlin, M.: Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 412–429. Springer, Heidelberg (2005)
7. Boldyreva, A., Fischlin, M.: On the security of OAEP. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 210–225. Springer, Heidelberg (2006)
8. Boneh, D.: Simplified OAEP for the RSA and rabin functions. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 275–291. Springer, Heidelberg (2001)
9. Brown, D.R.L.: What hashes make RSA-OAEP secure? Cryptology ePrint Archive, Report 2006/223 (2006), `http://eprint.iacr.org/`
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
11. Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions (preliminary version). In: STOC, pp. 131–140 (1998)

[6] It is not exactly that, as $f_{k,ek}$ is a random permutation, not a function.

12. Chevallier-Mames, B., Phan, D.H., Pointcheval, D.: Optimal asymmetric encryption and signature paddings. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 254–268. Springer, Heidelberg (2005)
13. Coron, J.-S., Joye, M., Naccache, D., Paillier, P.: Universal padding schemes for RSA. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 226–241. Springer, Heidelberg (2002)
14. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)
15. Damgård, I.: Collision free hash functions and public key signature schemes. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 203–216. Springer, Heidelberg (1988)
16. Dodis, Y., Freedman, M.J., Jarecki, S., Walfish, S.: Versatile padding schemes for joint signature and encryption. In: ACM CCS, pp. 344–353 (2004)
17. Dodis, Y., Oliveira, R., Pietrzak, K.: On the generic insecurity of the full domain hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
18. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. Journal of Cryptology 17(2), 81–104 (2004)
19. Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: FOCS, pp. 305–313 (2000)
20. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: FOCS, pp. 325–335 (2000)
21. Gertner, Y., Malkin, T.G., Myers, S.: Towards a separation of semantic and CCA security for public key encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 434–455. Springer, Heidelberg (2007)
22. Gertner, Y., Malkin, T., Reingold, O.: On the impossibility of basing trapdoor functions on trapdoor predicates. In: FOCS, pp. 126–135 (2001)
23. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC, pp. 25–32 (1989)
24. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: FOCS, pp. 102–115 (2003)
25. Haitner, I.: Implementing oblivious transfer using collection of dense trapdoor permutations. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 394–409. Springer, Heidelberg (2004)
26. Hsiao, C.-Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (2004)
27. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC, pp. 44–61 (1989)
28. Kim, J.H., Simon, D.R., Tetali, P.: Limits on the efficiency of one-way permutation-based hash functions. In: FOCS, pp. 535–542 (1999)
29. Kobara, K., Imai, H.: OAEP++: A very simple way to apply OAEP to deterministic OW-CPA primitives. Cryptology ePrint Archive, Report 2002/130 (2002), `http://eprint.iacr.org/`
30. Komano, Y., Ohta, K.: Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 366–382. Springer, Heidelberg (2003)
31. Lindell, Y.: A simpler construction of CCA2-secure public-key encryption under general assumptions. Journal of Cryptology 19(3), 359–377 (2006)

32. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: FOCS, pp. 120–130 (1999)
33. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC (1990)
34. Okamoto, T., Pointcheval, D.: REACT: Rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)
35. Paillier, P., Villar, J.L.: Trading one-wayness against chosen-ciphertext security in factoring-based encryption. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 252–266. Springer, Heidelberg (2006)
36. Phan, D.H., Pointcheval, D.: Chosen-ciphertext security without redundancy. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 1–18. Springer, Heidelberg (2003)
37. Phan, D.H., Pointcheval, D.: OAEP 3-round:A generic and secure asymmetric encryption padding. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 63–77. Springer, Heidelberg (2004)
38. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 433–444. Springer, Heidelberg (1993)
39. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 419–436. Springer, Heidelberg (2009)
40. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
41. Shoup, V.: OAEP reconsidered. Journal of Cryptology 15(4), 223–249 (2002)
42. Simon, D.R.: Findings Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)

Simulation without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters' IBE Scheme

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. Waters' variant of the Boneh-Boyen IBE scheme is attractive because of its efficency, applications, and security attributes, but suffers from a relatively complex proof with poor concrete security. This is due in part to the proof's "artificial abort" step, which has then been inherited by numerous derivative works. It has often been asked whether this step is necessary. We show that it is not, providing a new proof that eliminates this step. The new proof is not only simpler than the original one but offers better concrete security for important ranges of the parameters. As a result, one can securely use smaller groups, resulting in significant efficiency improvements.

1 Introduction

The importance of identity-based encryption (IBE) as a cryptographic primitive stems from its widespread deployment and the numerous applications enabled by it. Since the initial work on providing realizations of IBE [8,15], improving the efficiency, security, and extensibility of the fundamental primitive has consequently received substantial attention from the research community. A challenging problem has been to arrive at a practical IBE scheme with a tight security reduction under standard assumptions. (The most attractive target being DBDH without relying on random oracles.) While a typical approach for progressing towards this goal is proposing new constructions, in this paper we take another route: improving the concrete security of existing constructions. This requires providing better proofs of security and analyzing the impact of their tighter reductions.

Why concrete security? Informally speaking, consider an IBE scheme with a security reduction showing that attacking the scheme in time t with success probability ϵ implies breaking some believed-to-be hard problem in time $t + \omega_1$ with success probability $\epsilon' \geq \epsilon/\omega_2$. Tightness of the reduction refers to the value of ω_1 (the overhead in time needed to solve the hard problem using the scheme attacker) and of ω_2 (the amount by which the success probability decreases).

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 407–424, 2009.

Unlike asymptotic treatments, provably-secure IBE has a history of utilizing concrete security, meaning specifying ω_1 and ω_2 explicitly. Concrete-security for IBE started with Boneh and Franklin [8] and has been continued in subsequent works, e.g. [6,7,9,31,19,23] to name just a few.

As Gentry points out [19], concrete security and tight reductions are not just theoretical issues for IBE, rather they are of utmost practical import: the speed of implementations increases as ω_1 and/or ω_2 decrease. This is because security guarantees are lost unless the size of groups used to implement a scheme grow to account for the magnitude of these values. In turn group size dictates performance: exponentiations in a group whose elements can be represented in r bits takes roughly $\mathcal{O}(r^3)$ time. As a concrete example, this means that performing four 160-bit group exponentiations can be significantly faster than a *single* 256-bit group exponentiation. In practice even a factor of two efficiency slow-down is considered significant (let alone a factor of four), so finding as-tight-as-possible reductions is crucial.

Overview of IBE approaches. All practical IBE systems currently known are based on bilinear pairings. We can partition the space of such systems along two dimensions, as shown in the left table of Figure 1. In one dimension is whether one utilizes random oracles or not. In the other is the flavor of hard problem used, whether it be interactive (for example the q-BDHI assumption [6]) or non-interactive (for example bilinear Diffie-Hellman, or BDH, style assumptions [8]). Of note is that Katz and Wang [23], in the "random oracle/BDH" setting, and Gentry [19], in the "no random oracle/interactive setting", have essentially solved the problem of finding practical schemes with tight reductions. On the other hand, finding practical schemes with tight reductions in the "no random oracle/BDH" setting represents a hard open problem mentioned in numerous works [6,7,31,19]. This last setting turns out to be attractive for two reasons. First, from a security perspective, it is the most conservative (and consequently most challenging) with regard to choice of assumptions. Second, schemes thus far proposed in this setting follow a framework due to Boneh and Boyen [6] (so-called "commutative blinding") that naturally supports many valuable extensions: hierarchical IBE [22], attribute-based IBE [29], direct CCA-secure encryption [10,24], etc.

Progress in this setting is summarized in the right table of Figure 1. Boneh and Boyen initiated work here with the BB_1 scheme (the first scheme in [6]). They prove it secure under the decisional BDH (DBDH) assumption, but in the selective-ID attack model of [11] in which adversaries must commit to a targeted identity before seeing the IBE system's parameters. Boneh and Boyen show how to prove full security, but the reduction doing so is exponentially loose (briefly, because it requires guessing the hash of the to-be-attacked identity).

Waters' proposed a variant of BB_1 that we'll call Wa [31]. This variant requires larger public parameters, but can be proven fully secure with a polynomial reduction to DBDH that does not use random oracles. The relatively complex security proof relies on a novel "artificial abort" step, that, while clever, is

	Interactive	BDH
RO model	SK	BF , KW
Standard model	BB_2 , Ge	BB_1 , Wa

Scheme	Security	Reduction
BB_1	selective-ID	polynomial
BB_1	full	exponential
Wa	full	polynomial

Fig. 1. A comparison of practical IBE schemes. BF is the Boneh-Franklin scheme [8]; SK is the Sakai-Kasahara scheme [30]; KW is the Katz-Wang scheme [23]; BB_1 and BB_2 are the first and second Boneh-Boyen schemes from [6]; Wa is Waters' scheme [31]; and Ge is Gentry's scheme [19]. **(Left)** The assumptions (an interactive assumption versus bilinear Diffie-Hellman) and model (random oracles or not) used to prove security of the schemes. **(Right)** Types of security offered by standard model BDH-based systems and asymptotic reduction tightness.

unintuitive. It also significantly hurts the concrete security and, thereby, efficiency of the scheme. Many researchers in the community have asked whether artificial aborts can be dispensed with, but the general consensus seems to have been that the answer is "no" and that the technique is somehow fundamental to proving security. This folklore assessment (if true) is doubly unfortunate because Wa, inheriting the flexibility of the Boneh-Boyen framework, has been used in numerous diverse applications [10,1,5,27,12,13,20,24]. As observed in [24], some of these subsequent works offer difficult to understand (let alone verify) proofs, due in large part to their use of the artificial abort technique in a more-or-less black-box manner. They also inherit its concrete security overhead.

This paper. Our first contribution is to provide a novel proof of Waters' variant that completely eliminates the artificial abort step. The proof, which uses several new techniques and makes crucial use of code-based games [4], provides an alternate and (we feel) more intuitive and rigorous approach to proving the security of Wa. Considering the importance of the original proof (due to its direct or indirect use in [10,1,5,27,12,13,20,24]), a more readily understood proof is already a significant contribution. Our reduction (like Waters') is not tight, but as we see below it offers better concrete security for many important parameter choices, moving us closer to the goal of standard model BDH-based schemes with tight reductions. The many Waters'-derived works [10,1,5,27,12,13,20,24] inherit the improvements in concrete security. We briefly describe these derivatives in Appendix A.

We now have the BB_1 and Wa schemes, the former with an exponentially-loose reduction and the latter with now two polynomial reductions each having a complex concrete security formula. What is the most efficient approach for providing provably-secure DBDH-based IBE? Since we want to account for the impact of reduction tightness, answering this question requires work. We offer a framework for computing the concrete efficiency of reductions, adopting techniques from [26,25,18]. Efficiency is measured by mapping desired (provable) security levels to requisite group sizes. Not only does this approach provide a

κ	ϵ	q	s_{BB}	s_W	s_{BR}	$\mathbf{T}_{\mathsf{Enc}}(s_W)/\mathbf{T}_{\mathsf{Enc}}(s_{BR})$
60	2^{-20}	2^{20}	192	192	128	**9**
70	2^{-20}	2^{20}	256	192	128	**9**
80	2^{-30}	2^{30}	256	256	192	**5**
90	2^{-30}	2^{30}	–	256	192	**5**
100	2^{-10}	2^{10}	–	128	192	1/9
100	2^{-40}	2^{40}	–	256	192	**5**
192	2^{-40}	2^{40}	–	256	–	–

Fig. 2. Table showing the security level of the pairing setups required to achieve κ-bits of security for the BB_1 and Wa encryption schemes when adversaries achieve ϵ success probability using q key extraction queries. Loosely speaking, the security level of the pairing setup is $(\log p)/2$ where p is the size of the first pairing group. Here s_{BB}, s_W, s_{BR} are, respectively, the securities of the pairing setups for BB_1, Wa under Waters' reduction, and Wa under the new reduction. The final column represents the (approximate) ratio of encryption times for Wa as specified by the two reductions. A dash signifies that one needs a pairing setup of security greater than 256.

metric for comparing different reductions, it also allows comparing the resultant bit-operation speed of schemes when each is instantiated in groups of size sufficient to account for the reduction. Providing such a framework that simultaneously provides simplicity, accuracy, and fairness (i.e. not biased towards particular schemes/reductions) turned out to be very challenging.

Let us first mention the high-level results, before explaining more. In the end our framework implies that Waters' variant usually provides faster standard model encryption (than BB_1). Our new proof provides a better reduction for low to mid range security parameters, while Waters' reduction is tighter for higher security parameters. The new reduction in fact drastically improves efficiency in the former category, offering up to 9 times faster encryption for low parameters and 5 times faster encryption for mid-range security levels. Where Waters' reduction is tighter, we can continue to choose group size via it; the new reduction never *hurts* efficiency.

BB_1 does better than Wa when identities are short, such as $n = 80$ bits. We have, however, focused on providing IBE with arbitrary identity spaces, which provides the most versatility. Supporting long identities (e.g. email addresses such as `john.doe123@anonymous.com`) requires utilizing a collision-resistant hash function to compress identities. In this case, the birthday bound mandates that the bit length n of hash outputs be double the desired security level, and this affects the BB_1 scheme more due to its reduction being loose by a factor of 2^n.

Framework details. We present some results of applying our framework in Figure 2. Let us explain briefly what the numbers signify and how we derived them. (Details are in the full version of this paper [3].) By a *setup* we mean

groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ admitting a bilinear map $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The setup provides security s (bits) if the best known algorithms to solve the discrete logarithm (DL) problem take at least 2^s time in any of the three groups. We assume (for these estimates but not for the proof!) that the best algorithm for solving DBDH is solving DL in one of the groups. An important practical issue in pairings-based cryptography is that setups for arbitrary security are not known. Accordingly, we will restrict attention to values $s = 80$, 112, 128, 192, and 256, based on information from [28,25,26,16,17]. Now we take as our target that the IBE scheme should provide κ bits of security. By this we mean that any adversary making at most $q = 1/\epsilon$ **Extract** queries and having running time at most $\epsilon 2^\kappa$ should have advantage at most ϵ. For each scheme/reduction pair we can then derive the security s of the underlying pairing setup required to support the desired level of security. See Figure 2 for $\mathsf{BB_1}$ (s_{BB}), Wa under Waters' reduction (s_W), and under the new reduction (s_{BR}).

Other related work and open problems. Recently Hofheinz and Kiltz describe programmable hash functions [21]. Their main construction uses the same hash function (originally due to Chaum et al. [14]) as Waters', and they provide new proof techniques that provide a $\sqrt{n}$ (n is the length of identities) improvement on certain bounds that could be applicable to Wa. But this will only offer a small concrete security improvement compared to ours. Moreover, their results are asymptotic and hide (seemingly very large) unknown constants.

As mentioned, providing a scheme based on DBDH that has a tight security reduction (without random oracles) is a hard open problem, and one that remains after our work. (One reason we explain this is that we have heard it said that eliminating the artificial abort would solve the open problem just mentioned, but in fact the two seem to be unrelated.) Finding a tight reduction for Waters' (or another $\mathsf{BB_1}$-style scheme) is of particular interest since it would immediately give a hierarchical IBE (HIBE) scheme with security beyond a constant number of levels (the best currently achievable). From a practical point of view we contribute here, since better concrete security improves the (constant) number of levels achievable. From a theoretical perspective, this remains an open problem.

Viewing proofs as qualitative. We measure efficiency of schemes when one sets group size according to the best-known reduction. However, the fact that a proof implies the need for groups of certain size to guarantee security of the scheme does not mean the scheme is necessarily insecure (meaning there is an attack) over smaller groups. It simply means that the proof tells us nothing about security in these smaller groups. In the context of standards it is sometimes suggested one view a proof as a qualitative rather than quantitative guarantee, picking group sizes just to resist the best known attack. Our sense is that this procedure is not viewed as ideal even by its proposers but rather forced on them by the looseness of reductions. To rectify this gap, one must find tighter reductions, and our work is a step to this end.

2 Definitions and Background

NOTATION. We fix *pairing parameters* $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are groups of prime order p; $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate, efficiently computable bilinear map; and ψ: $\mathbb{G}_2 \to \mathbb{G}_1$ is an efficiently computable isomorphism [8]. Let $\mathsf{T}_{\exp}(\mathbb{G})$ denote the time to compute an exponentiation in a group $\mathbb{G}$. Similarly, let $\mathsf{T}_{\mathrm{op}}(\mathbb{G})$ denote the time to compute a group operation in a group $\mathbb{G}$. Let T_ψ denote the time to compute ψ. Let $\mathbb{G}^* = \mathbb{G} - \{\mathbf{1}\}$ denote the set of generators of $\mathbb{G}$ where $\mathbf{1}$ is the identity element of $\mathbb{G}$.

Vectors are written in boldface, e.g. $\mathbf{u} \in \mathbb{Z}_p^{n+1}$ is a vector of $n+1$ values each in $\mathbb{Z}_p$. We denote the i^{th} component of a vector $\mathbf{u}$ by $\mathbf{u}[i]$. If $S \in \{0,1\}^*$ then $|S|$ denotes its length and $S[i]$ denotes its i^{th} bit. For integers i, j we let $[i\,..\,j] = \{i, \ldots, j\}$. The running time of an adversary $\mathcal{A}$ is denoted $\mathbf{T}(\mathcal{A})$. We use big-oh notation with the understanding that this hides a small, fixed, machine-dependent constant.

GAMES. Our security definitions and proofs use code-based games [4], and so we recall some background from [4]. A game (look at Figure 3 for examples) has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary $\mathcal{A}$ as follows. First, **Initialize** executes, and its outputs are the inputs to $\mathcal{A}$. Then $\mathcal{A}$ executes, its oracle queries being answered by the corresponding procedures of G. When $\mathcal{A}$ terminates, its output becomes the input to the **Finalize** procedure. The output of the latter is called the output of the game, and we let $\mathrm{G}^{\mathcal{A}} \Rightarrow y$ denote the event that this game output takes value y. The boolean flag bad is assumed initialized to false. Games G_i, G_j are identical-until-bad if their code differs only in statements that follow the setting of bad to true. We let "$\mathrm{G}_i^{\mathcal{A}}$ sets bad" denote the event that game G_i, when executed with adversary $\mathcal{A}$, sets bad to true (and similarly for "$\mathrm{G}_i^{\mathcal{A}}$ doesn't set bad"). It is shown in [4] that if $\mathrm{G}_i, \mathrm{G}_j$ are identical-until-bad and $\mathcal{A}$ is an adversary, then

$$\Pr\left[\mathrm{G}_i^{\mathcal{A}} \text{ sets } \mathsf{bad}\right] = \Pr\left[\mathrm{G}_j^{\mathcal{A}} \text{ sets } \mathsf{bad}\right]. \tag{1}$$

The fundamental lemma of game-playing [4] says that if $\mathrm{G}_i, \mathrm{G}_j$ are identical-until-bad then for any y

$$\Pr\left[\mathrm{G}_i^{\mathcal{A}} \Rightarrow y\right] - \Pr\left[\mathrm{G}_j^{\mathcal{A}} \Rightarrow y\right] \leq \Pr\left[\mathrm{G}_i^{\mathcal{A}} \text{ sets } \mathsf{bad}\right].$$

This lemma is useful when the probability that bad is set is small, but in our setting this probability will be close to one. We will instead use the following variant:

Lemma 1. Let $\mathrm{G}_i, \mathrm{G}_j$ be identical-until-bad games and let $\mathcal{A}$ be an adversary. Then for any y

$$\Pr\left[\mathrm{G}_i^{\mathcal{A}} \Rightarrow y \wedge \mathrm{G}_i^{\mathcal{A}} \text{ doesn't set } \mathsf{bad}\right] = \Pr\left[\mathrm{G}_j^{\mathcal{A}} \Rightarrow y \wedge \mathrm{G}_j^{\mathcal{A}} \text{ doesn't set } \mathsf{bad}\right]. \ \square$$

Lemma 1 is implicit in the proof of the fundamental lemma of [4].

DBDH PROBLEM.The Decisional Bilinear Diffie-Hellman (DBDH) assumption (in the asymmetric setting) [6] is captured by the game described in Figure 3. We

Game $\mathrm{DBDH}_{\mathsf{GP}}$

proc. Initialize:
$g_2 \xleftarrow{\$} \mathbb{G}_2^* \,;\, g_1 \leftarrow \psi(g_2) \,;\, a, b, s \xleftarrow{\$} \mathbb{Z}_p \,;\, d \xleftarrow{\$} \{0,1\}$
If $d = 1$ then $W \xleftarrow{\$} \mathbf{e}(g_1, g_2)^{abs}$ Else $W \xleftarrow{\$} \mathbb{G}_T$
Ret $(g_1, g_2, g_2^a, g_2^b, g_2^s, W)$

proc. Finalize(d'):
Ret $(d' = d)$

Game $\mathrm{IND\text{-}CPA}_{\mathsf{IBE}}$

proc. Initialize:
$(mpk, msk) \xleftarrow{\$} \mathsf{Pg}$; $c \xleftarrow{\$} \{0,1\}$
Ret mpk

proc. LR(I, M_0, M_1):
Ret $\mathsf{Enc}(mpk, I, M_c)$

proc. Extract(I):
Ret $\mathsf{Kg}(mpk, msk, I)$

proc. Finalize(c'):
Ret $(c' = c)$

Fig. 3. The DBDH and IND-CPA games

define the dbdh-advantage of an adversary $\mathcal{A}$ against $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ by

$$\mathbf{Adv}^{\mathrm{dbdh}}_{\mathsf{GP}}(\mathcal{A}) = 2 \cdot \Pr\left[\, \mathrm{DBDH}^{\mathcal{A}}_{\mathsf{GP}} \Rightarrow \mathsf{true} \,\right] - 1 \,. \tag{2}$$

Identity-based encryption. An *identity-based encryption (IBE) scheme* is a tuple of algorithms $\mathsf{IBE} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with associated identity space $\mathsf{IdSp} \subseteq \{0,1\}^*$ and message space MsgSp. The key-issuing center runs the parameter generation algorithm Pg (which takes no input) generates a master public key mpk and a master secret key msk. The former is publicly distributed. The key generation algorithm Kg takes as input mpk, msk, I, where $I \in \mathsf{IdSp}$, and outputs a secret key sk for party I. The encryption algorithm Enc takes inputs mpk, I, M, where $I \in \mathsf{IdSp}$ and $M \in \mathsf{MsgSp}$, and outputs a ciphertext C. The deterministic decryption algorithm Dec takes inputs mpk, sk, I, C and outputs either $\perp$ or a plaintext M. We require the usual consistency, namely that $\mathsf{Dec}(mpk, sk, I, \mathsf{Enc}(mpk, I, M)) = M$ with probability one for all $I \in \mathsf{IdSp}$ and $M \in \mathsf{MsgSp}$, where the probability is over $(mpk, msk) \xleftarrow{\$} \mathsf{Pg}$; $sk \xleftarrow{\$} \mathsf{Kg}(mpk, msk, I)$ and the coins used by Enc. We use the notion of privacy from [8], namely indistinguishability under chosen-plaintext attack (ind-cpa). The ind-cpa advantage of an adversary $\mathcal{A}$ against an IBE scheme IBE is defined by

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{IBE}}(\mathcal{A}) = 2 \cdot \Pr\left[\, \mathrm{IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBE}} \Rightarrow \mathsf{true} \,\right] - 1 \,, \tag{3}$$

where game IND-CPA is shown in Figure 3. We only allow *legitimate* adversaries, where adversary $\mathcal{A}$ is legitimate if it makes only one query (I^*, M_0, M_1) to **LR**, for some $I^* \in \mathsf{IdSp}$ and $M_0, M_1 \in \mathsf{MsgSp}$ with $|M_0| = |M_1|$, and never queries I^* to **Extract**. Here $|M|$ denotes the length of some canonical string encoding of a message $M \in \mathsf{MsgSp}$. (In the schemes we consider messages are group elements.)

Waters' IBE scheme. Let n be a positive integer. Define the hash family $H\colon\ \mathbb{G}_1^{n+1} \times \{0,1\}^n \to \mathbb{G}_1$ by $H(\mathbf{u}, I) = \mathbf{u}[0] \prod_{i=1}^{n} \mathbf{u}[i]^{I[i]}$ for any $\mathbf{u} \in \mathbb{G}_1^{n+1}$ and any $I \in \{0,1\}^n$. The Waters IBE scheme $\mathsf{Wa} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ associated to GP and n has associated identity space $\mathsf{IdSp} = \{0,1\}^n$ and message space $\mathsf{MsgSp} = \mathbb{G}_T$, and its first three algorithms are as follows:

proc. Pg	**proc.** $\mathsf{Kg}(mpk, msk, I)$	**proc.** $\mathsf{Enc}(mpk, I, M)$
$A_1 \xleftarrow{\$} \mathbb{G}_1$; $g_2 \xleftarrow{\$} \mathbb{G}_2^*$	$(g_2, A_1, B_2, \mathbf{u}) \leftarrow mpk$	$(g_2, A_1, B_2, \mathbf{u}) \leftarrow mpk$
$b \xleftarrow{\$} \mathbb{Z}_p$; $B_2 \leftarrow g_2^b$; $\mathbf{u} \xleftarrow{\$} \mathbb{G}_1^{n+1}$	$K \leftarrow msk$; $r \xleftarrow{\$} \mathbb{Z}_p$	$s \xleftarrow{\$} \mathbb{Z}_p$
$mpk \leftarrow (g_2, A_1, B_2, \mathbf{u})$	Ret $(K \cdot H(\mathbf{u}, I)^r, g_2^r)$	$c_1 \leftarrow \mathbf{e}(A_1, B_2)^s \cdot M$
$msk \leftarrow A_1^b$		$(c_2, c_2) \leftarrow (g_2^s, H(\mathbf{u}, I)^s)$
Ret (mpk, msk)		Ret (c_1, c_2, c_3)

Above, when we write $(g_2, A_1, B_2, \mathbf{u}) \leftarrow mpk$ we mean mpk is parsed into its constituent parts. We do not specify the decryption algorithm since it is not relevant to IND-CPA security; it can be found in [31].

In [31] the scheme is presented in the symmetric setting where $\mathbb{G}_1 = \mathbb{G}_2$. While this makes notation simpler, we work in the asymmetric setting because it allows pairing parameters for higher security levels [17].

The hash function used by Waters' scheme has restricted domain. One can extend to $\mathsf{IdSp} = \{0,1\}^*$ by first hashing an identity with a collision-resistant hash function to derive an n-bit string. To ensure security from birthday attacks, the output length n of the CR function must have bit-length at least twice that of the desired security parameter.

WATERS' RESULT. Waters [31] proves the security of the Wa scheme associated to GP, n under the assumption that the DBDH problem in GP is hard. Specifically, let $\mathcal{A}$ be an ind-cpa adversary against Wa that runs in time at most t, makes at most $q \in [1\,..\,p/4n]$ queries to its **Extract** oracle and has advantage $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})$. Then [31, Theorem 1] presents a dbdh-adversary $\mathcal{B}_{\mathsf{Wa}}$ such that

$$\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}_{\mathsf{Wa}}) \geq \frac{\epsilon}{32(n+1)q} \text{ , and} \tag{4}$$

$$\mathbf{T}(\mathcal{B}_{\mathsf{Wa}}) = \mathbf{T}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n, q) + \mathbf{T}_{\text{abort}}(\epsilon, n, q) \tag{5}$$

where

$$\mathbf{T}_{\text{sim}}(n, q) = \mathcal{O}\left(\mathsf{T}_{\psi} + (n+q) \cdot \mathsf{T}_{\text{exp}}(\mathbb{G}_1) + q \cdot \mathsf{T}_{\text{exp}}(\mathbb{G}_2) + qn + \mathsf{T}_{\text{op}}(\mathbb{G}_T)\right) \tag{6}$$

$$\mathbf{T}_{\text{abort}}(\epsilon, n, q) = \mathcal{O}(q^2 n^2 \epsilon^{-2} \ln(\epsilon^{-1}) \ln(qn)) \,. \tag{7}$$

An important factor in the "looseness" of the reduction is the $\mathbf{T}_{\text{abort}}(\epsilon, n, q)$ term, which can be very large, making $\mathbf{T}(\mathcal{B}_{\mathsf{Wa}})$ much more than $\mathbf{T}(\mathcal{A})$. This term arises from the "artificial abort" step. (In [31], the $\mathbf{T}_{\text{abort}}(\epsilon, n, q)$ term only has a qn factor in place of the $q^2 n^2$ factor we show. However, the step requires performing q times up to n operations over the integers modulo $4q$ for each of the $\ell = \mathcal{O}(qn\epsilon^{-2} \ln \epsilon^{-1} \ln qn)$ vectors selected, so our term represents the actual cost.)

3 New Proof of Waters' IBE without Artificial Aborts

We start with some high-level discussion regarding Waters' original proof and the reason for the artificial abort. First, one would hope to specify a simulator

that, given IND-CPA adversary $\mathcal{A}$ that attacks the IBE scheme using q **Extract** queries and gains advantage ϵ, solves the DBDH problem with advantage not drastically worse than ϵ. But $\mathcal{A}$ can make **Extract** queries that force any conceivable simulator to fail, i.e. have to abort. This means that the advantage against DBDH is conditioned on $\mathcal{A}$ not causing an abort, and so it could be the case that $\mathcal{A}$ achieves ϵ advantage in the normal IND-CPA experiment but almost always causes aborts for the simulator. In this (hypothetical) case, the simulator could not effectively make use of the adversary, and the proof fails.

On the other hand, if one can argue that the lower and upper bounds on the probability of $\mathcal{A}$ causing an abort to occur are close (i.e. the case above does not occur), then the proof would go through. As Waters' points out [31], the natural simulator (that only aborts when absolutely necessarily) fails to provide such a guarantee. To compensate, Waters' introduced "artificial aborts". At the end of a successful simulation for the IBE adversary, the simulator $\mathcal{B}_{\mathsf{Wa}}$ used by Waters' generates $\mathcal{O}(qn\epsilon^{-2} \ln \epsilon^{-1} \ln qn)$ random vectors. These are used to estimate the probability that the **Extract** queries made by $\mathcal{A}$ cause an abort during any given execution of the simulator. The simulator then artificially aborts with some related probability. Intuitively, this forces the probability of aborting to be independent of $\mathcal{A}$'s particular queries. Waters' shows that $\mathcal{B}_{\mathsf{Wa}}$ provides the aforementioned guarantee of close lower and upper bounds and the proof goes through.

The artificial abort step seems strange because $\mathcal{B}_{\mathsf{Wa}}$ is forcing itself to fail even when it appears to have succeeded. The concrete security also suffers because the running time of the simulator goes up by $\mathbf{T}_{\text{abort}}(\epsilon, n, q)$ as shown in (7).

The rest of this section is devoted to proving the next theorem, which establishes the security of the Waters' IBE scheme without relying on an artificial abort step.

Theorem 1. Fix pairing parameters $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ and an integer $n \geq 1$, and let $\mathsf{Wa} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be the Waters IBE scheme associated to GP and n. Let $\mathcal{A}$ be an ind-cpa adversary against Wa which has advantage $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A}) > 0$ and makes at most $q \in [1\,..\,p\epsilon/9n]$ queries to its **Extract** oracle. Then there is a dbdh adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) \geq \frac{\epsilon^2}{27qn + 3\epsilon} \text{ , and} \tag{8}$$

$$\mathbf{T}(\mathcal{B}) = \mathbf{T}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n, q) \tag{9}$$

where $\mathbf{T}_{\text{sim}}(n, q)$ was defined by (6). □

The limitations on q—namely, $1 \leq q \leq p/4n$ in Waters' result and $1 \leq q \leq p\epsilon/9n$ in ours—are of little significance since in practice $p \geq 2^{160}$, $\epsilon \geq 2^{-80}$, and $n = 160$. For $q = 0$ there is a separate, tight reduction. The remainder of this section is devoted to the proof of Theorem 1.

Some definitions. Let $m = \lceil 9q/\epsilon \rceil$ and let $X = [-n(m-1)\,..\,0] \times [0\,..\,m-1] \times \cdots \times [0\,..\,m-1]$ where the number of copies of $[0\,..\,m-1]$ is n. For $\mathbf{x} \in X$,

$\mathbf{y} \in \mathbb{Z}_p^{n+1}$ and $I \in \{0,1\}^n$ we let

$$\mathsf{F}(\mathbf{x}, I) = \mathbf{x}[0] + \sum_{i=1}^{n} \mathbf{x}[i]I[i] \quad \text{and} \quad \mathsf{G}(\mathbf{y}, I) = \mathbf{y}[0] + \sum_{i=1}^{n} \mathbf{y}[i]I[i] \bmod p\,. \tag{10}$$

Note that while the computation of G above is over $\mathbb{Z}_p$, that of F is over $\mathbb{Z}$.

Adversary $\mathcal{B}$. Our DBDH adversary $\mathcal{B}$ is depicted in Figure 4, where the *simulation subroutines* KgS and EncS are specified below. There are two main differences between our adversary and that of Waters'. The first is that in our case the parameter m is $\mathcal{O}(q/\epsilon)$ while in Waters' case it is $\mathcal{O}(q)$. The second difference of course is that Waters' adversary $\mathcal{B}_{\mathsf{Wa}}$, unlike ours, includes the artificial abort step. Once $\mathcal{A}$ has terminated, this step selects $l = \mathcal{O}(qn\epsilon^{-2}\ln(\epsilon^{-1})\ln(qn))$ new random vectors $\mathbf{x}_1, \ldots, \mathbf{x}_l$ from X. Letting $I_1, \ldots, I_q$ denote the identities queried by $\mathcal{A}$ to its **Extract** oracle and I_0 the identity queried to the **LR** oracle, it then evaluates $\mathsf{F}(\mathbf{x}_i, I_j)$ for all $1 \le i \le l$ and $0 \le j \le q$, and uses these values to approximate the probability that bad is set. It then aborts with some related probability. Each computation of F takes $\mathcal{O}(n)$ time, and there are q such computations for each of the l samples, accounting for the estimate of (7). In addition there are some minor differences between the adversaries. For example, $\mathbf{x}$ is chosen differently. (In [31] it is taken from $[0\,..\,m-1]^{n+1}$, and an additional value $k \in [0\,..\,n]$, which we do not have, is mixed in.)

We note that our adversary in fact *never* aborts. Sometimes, it is clearly returning incorrect answers (namely $\perp$) to $\mathcal{A}$'s queries. Adversary $\mathcal{A}$ will recognize this, and all bets are off as to what it will do. Nonetheless, $\mathcal{B}$ continues the execution of $\mathcal{A}$. Our analysis will show that $\mathcal{B}$ has the claimed properties regardless.

An analysis of the running time of $\mathcal{B}$, justifying equations (6) and (9), is given in the full version of the paper [3].

Simulation subroutines. We define the subroutines that $\mathcal{B}$ utilizes to answer **Extract** and **LR** queries. We say that $(g_1, g_2, A_2, A_1, B_2, B_1, \mathbf{x}, \mathbf{y}, \mathbf{u}, S, W)$ are *simulation parameters* if: $g_2 \in \mathbb{G}_2^*$; $g_1 = \psi(g_2) \in \mathbb{G}_1^*$; $A_2 \in \mathbb{G}_2$; $A_1 = \psi(A_2) \in \mathbb{G}_1$; $B_2 \in \mathbb{G}_2$; $B_1 = \psi(B_2) \in \mathbb{G}_1$; $\mathbf{x} \in X$; $\mathbf{y} \in \mathbb{Z}_p^{n+1}$; $\mathbf{u}[j] = B_1^{\mathbf{x}[j]} g_1^{\mathbf{y}[j]}$ for $j \in [0\,..\,n]$; $S \in \mathbb{G}_2$; and $W \in \mathbb{G}_T$. We define the following procedures:

proc. $\mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$

$r \xleftarrow{\$} \mathbb{Z}_p$; $w \leftarrow \mathsf{F}(\mathbf{x}, I)^{-1} \bmod p$

$L_1 \leftarrow B_1^{\mathsf{F}(\mathbf{x},I)\cdot r} g_1^{\mathsf{G}(\mathbf{y},I)\cdot r} A_1^{-\mathsf{G}(\mathbf{y},I)w}$

$L_2 \leftarrow g_2^r A_2^{-w}$

Ret (L_1, L_2)

proc. $\mathsf{EncS}(S, W, M, \mathbf{y}, I)$

$C_1 \leftarrow W \cdot M$

$C_2 \leftarrow S$; $C_3 \leftarrow \psi(S)^{\mathsf{G}(\mathbf{y},I)}$

Ret (C_1, C_2, C_3)

Note that if $\mathsf{F}(\mathbf{x}, I) \neq 0$ then $\mathsf{F}(\mathbf{x}, I) \not\equiv 0 \pmod p$ so the quantity w computed by KgS is well-defined whenever $\mathsf{F}(\mathbf{x}, I) \neq 0$. This is because the absolute value of $\mathsf{F}(\mathbf{x}, I)$ is at most

$$n(m-1) = n\left(\left\lceil \frac{9q}{\epsilon} \right\rceil - 1\right) < \frac{9nq}{\epsilon} \le p\,, \tag{11}$$

Adversary $\mathcal{B}(g_1, g_2, A_2, B_2, S, W)$:
$c \xleftarrow{\$} \{0,1\}$; $A_1 \leftarrow \psi(A_2)$; $B_1 \leftarrow \psi(B_2)$
For $j = 0, \ldots, n$ do
  $\mathbf{y}[j] \xleftarrow{\$} \mathbb{Z}_p$
  If $j = 0$ then $\mathbf{x}[j] \xleftarrow{\$} [-n(m-1) \,..\, 0]$
  Else $\mathbf{x}[j] \xleftarrow{\$} [0 \,..\, m-1]$
  $\mathbf{u}[j] \leftarrow B_1^{\mathbf{x}[j]} g_1^{\mathbf{y}[j]}$
$mpk \leftarrow (g_2, A_1, B_2, \mathbf{u})$
Run $\mathcal{A}(mpk)$, answering queries by

query Extract(I):
$sk(I) \leftarrow \perp$
If $\mathsf{F}(\mathbf{x}, I) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else $sk(I) \xleftarrow{\$} \mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$
Ret $sk(I)$

query LR(I, M_0, M_1):
$C \leftarrow \perp$
If $\mathsf{F}(\mathbf{x}, I) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else $C \leftarrow \mathsf{EncS}(S, W, M_c, \mathbf{y}, I)$
Ret C

$\mathcal{A}$ finishes, returning bit c'
If $\mathsf{bad} = \mathsf{true}$ then $c' \xleftarrow{\$} \{0,1\}$
If $c = c'$ then Ret 1 else Ret 0

Fig. 4. Adversary $\mathcal{B}$

the last because of the restriction on q in the theorem statement. The next lemma captures two facts about the simulation subroutines, which we will use in our analysis.

Lemma 2. Let $(g_1, g_2, A_2, A_1, B_2, B_1, \mathbf{x}, \mathbf{y}, \mathbf{u}, S, W)$ be simulation parameters. Let $I \in \{0,1\}^n$. Let $mpk = (g_2, A_1, B_2, \mathbf{u})$. Let b be the discrete log of B_1 to base g_1 and let $msk = A_1^b$. Let s be the discrete log of S to base g_2. Then if $\mathsf{F}(\mathbf{x}, I) \neq 0$ the outputs of $\mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$ and $\mathsf{Kg}(mpk, msk, I)$ are identically distributed. Also if $\mathsf{F}(\mathbf{x}, I) = 0$ then for any $M \in \mathsf{MsgSp}$, the output of $\mathsf{EncS}(S, W, M, \mathbf{y}, I)$ is $(W \cdot M, S, H(\mathbf{u}, I)^s)$. □

The proof of Lemma 2, which follows arguments given in [31], is given in the full version [3].

Overview. Consider executing $\mathcal{B}$ in game $\mathrm{DBDH}_{\mathsf{GP}}$. If $d = 1$, then Lemma 2 implies that adversary $\mathcal{B}$ correctly answers oracle queries as long as it does not set bad. On the other hand if $d = 0$ then $\mathcal{B}$'s output is a random bit. Attempting to conclude by showing that bad is seldom set fails, however, because in fact it will be set with probability close to 1. Alternatively, if one could show that the setting of bad is independent of the correctness of $\mathcal{B}$'s output, then one could

proc. Initialize: Game G_4

400 $A_1 \xleftarrow{\$} \mathbb{G}_1$; $g_2 \xleftarrow{\$} \mathbb{G}_2^*$; $b, s \xleftarrow{\$} \mathbb{Z}_p$; $i \leftarrow 0$
401 $B_2 \leftarrow g_2^b$; $S \leftarrow g_2^s$; $c, d \xleftarrow{\$} \{0,1\}$; $K \leftarrow A_1^b$
402 For $j = 0, \ldots, n$ do
403 $\quad \mathbf{z}[j] \xleftarrow{\$} \mathbb{Z}_p$; $\mathbf{u}[j] \leftarrow g^{\mathbf{z}[j]}$
404 $mpk \leftarrow (g, A_1, B_2, \mathbf{u})$
405 If $d = 1$ then $W \leftarrow \mathbf{e}(A_1, B_2)^s$
406 Else $W \xleftarrow{\$} \mathbb{G}_T$
407 Ret mpk

proc. Extract(I): Games G_3, G_4

320 $cnt \leftarrow cnt + 1$; $I_{cnt} \leftarrow I$
321 $r \xleftarrow{\$} \mathbb{Z}_p$; Ret $sk(I) \leftarrow (K \cdot H(\mathbf{u}, I)^r, g_2^r)$

proc. LR(I, M_0, M_1): Games G_3, G_4

330 $I_0 \leftarrow I$
331 Ret $C \leftarrow (W \cdot M_c, S, H(\mathbf{u}, I)^s)$

proc. Finalize(c'): Game G_4

440 For $j = 0, \ldots, n$ do
441 $\quad$ If $j = 0$ then $\mathbf{x}[j] \xleftarrow{\$} [-n(m-1) \,..\, 0]$
442 $\quad$ Else $\mathbf{x}[j] \xleftarrow{\$} [0 \,..\, m-1]$
443 For $j = 1, \ldots, cnt$ do
444 $\quad$ If $\mathsf{F}(\mathbf{x}, I_j) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
445 If $\mathsf{F}(\mathbf{x}, I_0) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
446 If $c = c'$ then Ret 1 else Ret 0

Fig. 5. The game G_4

conclude by multiplying the probabilities of these events. The difficulty in the proof is that this independence does not hold. Waters' artificial abort step is one way to compensate. However, we have dropped this (expensive) step and propose nonetheless to push an argument through. We will first use the game sequence G_0–G_4 to arrive at a game where the choice of $\mathbf{x}$ is independent of the game output. The subtle point is that this *still* does not provide independence between setting bad and the game output, because the identities chosen by $\mathcal{A}$ for its oracle queries affect both events. The first step in addressing this is a conditioning argument based on Lemma 4 which allows us to express a lower bound on the advantage of $\mathcal{B}$ in terms of probabilities $\gamma(\mathbf{I})$ associated to different queried identities. The crucial insight is that Lemma 5 gives upper and lower bounds on these probabilities that are very close, specifically within a factor of $1 - \epsilon$ of each other, due to our choice of $m = \mathcal{O}(q/\epsilon)$ rather than merely the $m = \mathcal{O}(q)$ of [31]. Using this allows us to conclude easily.

The game playing sequence. Assume without loss of generality that $\mathcal{A}$ always makes exactly q queries to its **Extract** oracle rather than at most q. The proof starts using a sequence of games $\mathrm{G}_0 - \mathrm{G}_4$ to move from $\mathcal{B}$ running in the DBDH

experiment to a game G_4 (shown in Figure 5) that is essentially the IND-CPA experiment, though with some additional bookkeeping. This transition is critical since it moves to a setting where the choice of $\mathbf{x}$ is clearly independent of $\mathcal{A}$'s choices. For brevity, we capture this game playing sequence via the following lemma whose proof is given in the full version [3]. Let GD_4 denote the event that $\mathrm{G}_4^{\mathcal{A}}$ does not set bad.

Lemma 3. $\mathbf{Adv}^{\mathrm{dbdh}}_{\mathsf{GP}}(\mathcal{B}) = 2 \cdot \Pr\left[\, \mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \,\right] - \Pr\left[\, \mathsf{GD}_4 \,\right]$ □

We now reach a subtle point. Consider the following argument: "The event GD_4 depends only on $\mathbf{x}$, which is chosen at lines 440–442 *after* the adversary and game outputs are determined. So GD_4 is independent of the event that $\mathrm{G}_4^{\mathcal{A}} \Rightarrow d$." If we buy this, the probability of the conjunct in Lemma 3 becomes the product of the probability of the constituent events, and it is quite easy to conclude. However, the problem is that the argument in quotes above is wrong. The reason is that GD_4 also depends on $I_0, \ldots, I_q$ and these are adversary queries whose values are not independent of the game output. Waters' compensates for this via the artificial abort step, but we do not have this step in $\mathcal{B}$ and propose to complete the analysis anyway.

Conditional independence lemma. Let

$$\mathsf{ID} = \{(I_0, \ldots, I_q) \in (\{0,1\}^n)^{q+1} \ : \ \forall i \in [1\,..\,q]\ (I_0 \neq I_i)\}\,.$$

For $(I_0, \ldots, I_q) \in \mathsf{ID}$ let

$$\gamma(I_0, \ldots, I_q) = \Pr\left[\, \mathsf{F}(\mathbf{x}, I_0) = 0 \wedge \mathsf{F}(\mathbf{x}, I_1) \neq 0 \wedge \cdots \wedge \mathsf{F}(\mathbf{x}, I_q) \neq 0 \,\right]$$

where the probability is taken over $\mathbf{x} \xleftarrow{\$} X$. This is the probability of GD_4 under a particular sequence of queried identities $I_0, \ldots, I_q$. (We stress that here we first fix $I_0, \ldots, I_q$ and then choose $\mathbf{x}$ at random.) If $\gamma(I_0, \ldots, I_q)$ were the same for all $(I_0, \ldots, I_q) \in \mathsf{ID}$ then the problem discussed above would be resolved. The difficulty is that $\gamma(I_0, \ldots, I_q)$ varies with $I_0, \ldots, I_q$. Our next lemma is the main tool to resolve the independence problem. Roughly it says that if we consider the conditional space obtained by conditioning on a particular sequence $I_0, \ldots, I_q$ of queried identities, then independence does hold. To formalize this, let $\mathsf{Q}(\mathbf{I})$ be the event that the execution of G_4 with $\mathcal{A}$ results in the identities $I_0, \ldots, I_q$ being queried by $\mathcal{A}$, where $\mathbf{I} = (I_0, \ldots, I_q)$. Then:

Lemma 4. For any $\mathbf{I} \in \mathsf{ID}$,

$$\Pr\left[\, \mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I}) \,\right] = \gamma(\mathbf{I}) \cdot \Pr\left[\, \mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I}) \,\right] \tag{12}$$

$$\Pr\left[\, \mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I}) \,\right] = \gamma(\mathbf{I}) \cdot \Pr\left[\, \mathsf{Q}(\mathbf{I}) \,\right] \tag{13}$$

□

Proof. The set of coin tosses underlying the execution of G_4 with $\mathcal{A}$ can be viewed as a cross product $\Omega = \Omega' \times X$, meaning each member ω of Ω is a pair $\omega = (\omega', \mathbf{x})$ where $\mathbf{x}$ is the choice made at lines 440–442 and ω' is all the rest of the game and adversary coins. For any $\mathbf{I} \in \mathsf{ID}$ let $\Omega'(\mathbf{I})$ be the set of all $\omega \in \Omega'$ such that the execution with ω produces $\mathbf{I}$ as the sequence of queried

identities. (Which $\mathbf{I}$ is produced depends only on ω' since $\mathbf{x}$ is chosen after $\mathcal{A}$ has terminated.) Let Ω'_{out} be the set of all $\omega \in \Omega'$ on which the execution outputs d. (Again, this is determined only by ω' and not $\mathbf{x}$.) Let $X_{\text{gd}}(\mathbf{I})$ be the set of all $\mathbf{x} \in X$ such that

$$\mathsf{F}(\mathbf{x}, I_0) = 0 \wedge \mathsf{F}(\mathbf{x}, I_i) \neq 0 \wedge \cdots \wedge \mathsf{F}(\mathbf{x}, I_q) \neq 0\,,$$

where $\mathbf{I} = (I_0, \ldots, I_q)$. Now observe that the set of coins leading to $\mathrm{G}_4^{\mathcal{A}} \Rightarrow d$ is $\Omega'_{\text{out}} \times X$ and the set of coins leading to $\mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})$ is $\Omega'(\mathbf{I}) \times X_{\text{gd}}(\mathbf{I})$. So

$$\begin{aligned}
\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right] &= \frac{|(\Omega'_{\text{out}} \times X) \cap (\Omega'(\mathbf{I}) \times X_{\text{gd}}(\mathbf{I}))|}{|\Omega' \times X|} \\
&= \frac{|(\Omega'_{\text{out}} \cap \Omega'(\mathbf{I})) \times X_{\text{gd}}(\mathbf{I})|}{|\Omega' \times X|} = \frac{|\Omega'_{\text{out}} \cap \Omega'(\mathbf{I})| \cdot |X_{\text{gd}}(\mathbf{I})|}{|\Omega'| \cdot |X|} \\
&= \frac{|\Omega'_{\text{out}} \cap \Omega'(\mathbf{I})| \cdot |X|}{|\Omega'| \cdot |X|} \cdot \frac{|X_{\text{gd}}(\mathbf{I})|}{|X|} = \frac{|(\Omega'_{\text{out}} \cap \Omega'(\mathbf{I})) \times X|}{|\Omega' \times X|} \cdot \frac{|X_{\text{gd}}(\mathbf{I})|}{|X|}\,.
\end{aligned}$$

But the first term above is $\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I})\,\right]$ while the second is $\gamma(\mathbf{I})$, establishing (12). For (13) we similarly have

$$\begin{aligned}
\Pr\left[\,\mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right] &= \frac{|\Omega'(\mathbf{I}) \times X_{\text{gd}}(\mathbf{I})|}{|\Omega' \times X|} = \frac{|\Omega'(\mathbf{I})|}{|\Omega'|} \cdot \frac{|X_{\text{gd}}(\mathbf{I})|}{|X|} \\
&= \frac{|\Omega'(\mathbf{I})| \cdot |X|}{|\Omega'| \cdot |X|} \cdot \frac{|X_{\text{gd}}(\mathbf{I})|}{|X|} = \frac{|\Omega'(\mathbf{I}) \times X|}{|\Omega' \times X|} \cdot \frac{|X_{\text{gd}}(\mathbf{I})|}{|X|}\,.
\end{aligned}$$

But the final terms above are $\Pr\left[\,\mathsf{Q}(\mathbf{I})\,\right]$ and $\gamma(\mathbf{I})$, respectively, establishing (13).

Analysis continued. Let $\gamma_{\min}$ be the smallest value of $\gamma(I_0, \ldots, I_q)$ taken over all $(I_0, \ldots, I_q) \in \mathsf{ID}$. Let $\gamma_{\max}$ be the largest value of $\gamma(I_0, \ldots, I_q)$ taken over all $(I_0, \ldots, I_q) \in \mathsf{ID}$. Using Lemma 3 we have that

$$\begin{aligned}
\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &= 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4\,\right] - \Pr\left[\,\mathsf{GD}_4\,\right] \\
&= \sum_{\mathbf{I} \in \mathsf{ID}} 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right] - \sum_{\mathbf{I} \in \mathsf{ID}} \Pr\left[\,\mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right]
\end{aligned}$$

and applying Lemma 4:

$$\begin{aligned}
\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &= \sum_{\mathbf{I} \in \mathsf{ID}} 2\gamma(\mathbf{I}) \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I})\,\right] - \sum_{\mathbf{I} \in \mathsf{ID}} \gamma(\mathbf{I}) \cdot \Pr\left[\,\mathsf{Q}(\mathbf{I})\,\right] \\
&\geq \gamma_{\min} \underbrace{\sum_{\mathbf{I} \in \mathsf{ID}} 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I})\,\right]}_{= 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d\,\right]} - \gamma_{\max} \underbrace{\sum_{\mathbf{I} \in \mathsf{ID}} \Pr\left[\,\mathsf{Q}(\mathbf{I})\,\right]}_{= 1} \\
&\geq 2\gamma_{\min} \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d\,\right] - \gamma_{\max}\,. \qquad (14)
\end{aligned}$$

Now

$$\begin{aligned}\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d\,\right] &= \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1 \mid d{=}1\,\right]\Pr\left[\,d{=}1\,\right] + \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 0 \mid d{=}0\right]\Pr\left[\,d{=}0\right] \\ &= \frac{1}{2}\cdot\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1 \mid d=1\,\right] + \frac{1}{2}\cdot\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 0 \mid d=0\,\right] \\ &= \frac{1}{2}\cdot\left(\frac{1}{2}+\frac{1}{2}\cdot\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})\right)+\frac{1}{2}\cdot\frac{1}{2} \qquad (15) \\ &= \frac{1}{4}\cdot\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})+\frac{1}{2} \qquad (16)\end{aligned}$$

where we justify (15) as follows. In the case that $d = 0$, the value W is uniformly distributed over $\mathbb{G}_T$ and hence line 331 gives $\mathcal{A}$ no information about the bit c. So the probability that $c = c'$ at line 446 is $1/2$. On the other hand if $d = 1$ then G_4 implements the IND-CPA$_{\mathsf{Wa}}$ game, so $2\cdot\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1 \mid d=1\,\right]-1 = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})$ by (3). We substitute (16) into (14) and get

$$\begin{aligned}\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &\geq 2\gamma_{\min}\left(\frac{1}{4}\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})+\frac{1}{2}\right)-\gamma_{\max} \\ &= \frac{\gamma_{\min}}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})+(\gamma_{\min}-\gamma_{\max})\,. \qquad (17)\end{aligned}$$

To finish the proof, we use the following:

Lemma 5. $\dfrac{1}{n(m-1)+1}\left(1-\dfrac{q}{m}\right) \leq \gamma_{\min} \leq \gamma_{\max} \leq \dfrac{1}{n(m-1)+1}$ □

The proof of Lemma 5, based on ideas in [31], is given in the full version of the paper [3]. Let $\alpha = 1/(n(m-1)+1)$. Recall that $m = \lceil 9q/\epsilon \rceil \geq 9q/\epsilon$ where $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})$. Then, applying Lemma 5 to (17) we get

$$\begin{aligned}\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &\geq \frac{\alpha}{2}\left(1-\frac{q}{m}\right)\epsilon+\alpha\left(1-\frac{q}{m}\right)-\alpha \;=\; \alpha\left[\frac{1}{2}\left(1-\frac{q}{m}\right)\epsilon-\frac{q}{m}\right] \\ &\geq \alpha\left[\frac{1}{2}\left(1-\frac{q\epsilon}{9q}\right)\epsilon-\frac{q\epsilon}{9q}\right] \;=\; \frac{\alpha\epsilon}{18}(7-\epsilon) \\ &\geq \frac{\alpha\epsilon}{3}\,. \qquad (18)\end{aligned}$$

Inequality (18) is justified by the fact that $\epsilon \leq 1$. Using the fact that $m = \lceil 9q/\epsilon \rceil \leq 9q/\epsilon + 1$ and substituting in for α, we complete the derivation of our lower bound for $\mathcal{B}$:

$$\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) \geq \frac{\epsilon}{3}\cdot\frac{1}{n(m-1)+1} \geq \frac{\epsilon}{3}\cdot\frac{1}{n(9q/\epsilon)+1} = \frac{\epsilon^2}{27qn+3\epsilon}\,.$$

Acknowledgments

We thank Brent Waters for pointing out a bug in the proof of an earlier version of this paper. We thank Sarah Shoup for participating in early stages of this

work. We thank Dan Boneh and Xavier Boyen for comments on earlier drafts of this work. This work was supported in part by NSF grants CNS 0524765 and CNS 0627779 and a gift from Intel corporation.

References

1. Abdalla, M., Catalano, D., Dent, A.W., Malone-Lee, J., Neven, G., Smart, N.P.: Identity-based encryption gone wild. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 300–311. Springer, Heidelberg (2006)
2. Abdalla, M., Kiltz, E., Neven, G.: Generalized key delegation for hierarchical identity-based encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 139–154. Springer, Heidelberg (2007)
3. Bellare, M., Ristenpart, T.: Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme (full version of this paper). Available from authors' home pages (January 2009)
4. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
5. Birkett, J., Dent, A.W., Neven, G., Schuldt, J.C.N.: Efficient chosen-ciphertext secure identity-based encryption with wildcards. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 274–292. Springer, Heidelberg (2007)
6. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Boyen, X.: General ad hoc encryption from exponent inversion IBE. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 394–411. Springer, Heidelberg (2007)
10. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM CCS 2005, November 7–11, pp. 320–329. ACM Press, New York (2005)
11. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
12. Chatterjee, S., Sarkar, P.: Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 424–440. Springer, Heidelberg (2006)
13. Chatterjee, S., Sarkar, P.: HIBE with short public parameters without random oracle. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 145–160. Springer, Heidelberg (2006)
14. Chaum, D., Evertse, J.-H., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)

15. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
16. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372 (2007), http://eprint.iacr.org/
17. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006), http://eprint.iacr.org/
18. Galindo, D.: The exact security of pairing based encryption and signature schemes. In: Based on a talk at Workshop on Provable Security, INRIA, Paris (2004), http://www.dgalindo.es/galindoEcrypt.pdf
19. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
20. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
21. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
22. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
23. Katz, J., Wáng, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM CCS 2003, October 27–30, pp. 155–164. ACM Press, New York (2003)
24. Kiltz, E., Galindo, D.: Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 336–347. Springer, Heidelberg (2006)
25. Lenstra, A.K.: Unbelievable security: Matching AES security using public key systems. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 67–86. Springer, Heidelberg (2001)
26. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. Journal of Cryptology: the journal of the International Association for Cryptologic Research 14(4), 255–293 (2001)
27. Naccache, D.: Secure and practical identity-based encryption. Cryptology ePrint Archive, Report 2005/369 (2005), http://eprint.iacr.org/
28. National Institute for Standards and Technology. Recommendation for Key Management Part 1: General (revised). NIST Special Publication 800-57 (2005)
29. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Sakai, R., Kasahara, M.: Id based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054 (2003), http://eprint.iacr.org/
31. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A Derivatives of Waters' IBE

A large body of research [10,1,5,24,27,12,13,20] utilizes Waters' scheme. Recall that Waters' already proposed a heirarchical IBE scheme based on Wa in [31], and subsequently there have been numerous derivative works. All use the artificial abort, either due to a black-box reduction to Waters' (H)IBE or as an explicit step in a direct proof. Our new proof technique immediately benefits those schemes that utilize Waters' scheme directly (i.e. in a black-box manner). For the rest, we believe that our techniques can be applied but have not checked the details.

- Naccache [27] and Chatterjee and Sarkar [12,13] independently and concurrently introduced a space-time trade-off for Wa that involves modifying the hash function utilized from $H(\mathbf{u}, I) = \mathbf{u}[0] \prod_{i=1}^{n} \mathbf{u}[i]^{I[i]}$ for $\mathbf{u} \in \mathbb{G}_1^{n+1}$ to $H'(\mathbf{u}, I) = \mathbf{u}[0] \prod_{i=1}^{\ell} \mathbf{u}[i]^{I[i]}$ where $\mathbf{u} \in \mathbb{G}_1^{\ell+1}$ and each $I[i]$ is now an n/ℓ-bit string. For appropriate choice of ℓ this will significantly reduce the number of elements included in the master public key. However the new choice of hash function impacts the reduction tightness, and since their proof includes just minor changes to Waters', our new reduction will increase the efficiency of this time/space trade-off for various security levels.
- In [10] BB_1- and Wa-based constructions of CCA-secure public-key encryption schemes and their proofs for the Wa case directly utilize artificial aborts.
- Kiltz and Galindo [24] propose a construction of CCA-secure identity-based key encapsulation that is a modified version of Wa.
- Wildcard IBE [1,5] is a generalization of heirarchical IBE that allows encryption to identities that include wildcards, e.g. "*@anonymous.com". In [1] a wildcard IBE scheme is proposed that utilizes the Waters HIBE scheme, and the proof is black-box to it. In [5] a wildcard identity-based KEM is produced based (in a non-black-box manner) on Waters' IBE.
- Wicked IBE [2] allows generation of private keys for wildcard identities. These private keys can then be used to generate derivative keys that replace the wildcards with any concrete identity string. They suggest using the Waters' HIBE scheme to achieve full security in their setting.
- Blind IBE, as introduced by Green and Hohenberger [20], enables the "trusted" master key generator to generate a private key for an identity without learning anything about the identity. To prove a Waters'-based blind IBE scheme secure they utilize the Naccache proof [27] (mentioned above). They utilize blind IBE schemes to build efficient and fully-simulatable oblivious transfer protocols based on the non-interactive assumptions inherited from the BDH-based IBE schemes used.

On the Portability of Generalized Schnorr Proofs

Jan Camenisch[1,⋆], Aggelos Kiayias[2,⋆⋆], and Moti Yung[3]

[1] IBM Research, Zurich, Switzerland
jca@zurich.ibm.com
[2] Computer Science and Engineering, University of Connecticut
Storrs, CT, USA
aggelos@cse.uconn.edu
[3] Google Inc. and Computer Science, Columbia University
New York, NY, USA
moti@cs.columbia.edu

Abstract. The notion of Zero Knowledge Proofs (of knowledge) [ZKP] is central to cryptography; it provides a set of security properties that proved indispensable in concrete protocol design. These properties are defined for any given input and also for any auxiliary verifier private state, as they are aimed at any use of the protocol as a subroutine in a bigger application. Many times, however, moving the theoretical notion to practical designs has been quite problematic. This is due to the fact that the most efficient protocols fail to provide the above ZKP properties *for all* possible inputs and verifier states. This situation has created various problems to protocol designers who have often either introduced imperfect protocols with mistakes or with lack of security arguments, or they have been forced to use much less efficient protocols in order to achieve the required properties. In this work we address this issue by introducing the notion of "protocol portability," a property that identifies input and verifier state distributions under which a protocol becomes a ZKP when called as a subroutine in a sequential execution of a larger application. We then concentrate on the very efficient and heavily employed "Generalized Schnorr Proofs" (GSP) and identify the portability of such protocols. We also point to previous protocol weaknesses and errors that have been made in numerous applications throughout the years, due to employment of GSP instances while lacking the notion of portability (primarily in the case of unknown order groups). This demonstrates that cryptographic application designers who care about efficiency need to consider our notion carefully. We provide a compact specification language for GSP protocols that protocol designers can employ. Our specification language is consistent with the ad-hoc notation that is currently widely used and it offers automatic derivation of the proof protocol while dictating its portability (i.e., the proper initial state and inputs) and its security guarantees. Finally, as a second alternative to designers wishing to use GSPs, we present a modification of GSP protocols that is unconditionally portable (i.e., ZKP) and is still quite efficient. Our constructions are the first such protocols proven secure in the standard model (as opposed to the random oracle model).

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

⋆ This research has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 216483.

⋆⋆ This research was partly supported by NSF CAREER 0447808, and NSF CNS 0831306.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 425–442, 2009.

1 Introduction

Motivation. Zero knowledge proofs [28] [ZKP], and zero knowledge proofs and arguments of knowledge in particular, are a central tool in cryptosystem and protocol design. These tools allow a designer to enforce parties to assure others that they take specified actions consistent with their internal knowledge state [26]. Properties of ZKP are defined over *all inputs* i.e., they provide security and correctness properties independently of input distribution. A shortcoming of ZKP's is that depending on the underlying language it can be hard to come up with efficient protocols. This has lead to the design of specialized protocols for specific language classes that occur often in applications. A celebrated example that has proven to be very useful in the design of efficient cryptographic schemes is known as Generalized Schnorr Proofs (extending the original seminal proof [37] to various algebraic settings like unknown order modular groups that arise in the context of the RSA cryptosystem). These protocols are at the heart of many efficient cryptographic systems and have been employed in a great number of schemes including: anonymous e-cash, anonymous voting, group signatures, distributed signing, distributed decryption, verifiable encryption, fair exchange, ring signatures, and credential systems. These schemes capitalized on the high efficiency of Schnorr's method and constitute, perhaps, the most extensive application of zero knowledge theory to practice so far. Further, a shorthand notation introduced in [14,15] for GSP has been extensively employed in the past and contributed to the wide employment of these protocols in cryptographic design. This notation suggested using e.g., $\mathsf{PK}(\alpha : y = g^{\alpha})$ to denote a proof of the discrete logarithm $\log_g y$ and it appeared in many works to describe quite complex discrete logarithm based relations, e.g., [3,7,8,9,10,11,13,24,25,30,31,32,33,34,38,39,40,41,42,43]. What has been often overlooked though is the fact that Generalized Schnorr Proofs are *not* zero-knowledge proofs of knowledge! This is a consequence of the fact that the security properties of such protocols are affected by the input distribution of the involved parties. Interestingly, despite the long line of works in the proper formalization of zero-knowledge proofs, this aspect has been largely overlooked, mainly due to the fact that it is only critical from an application-oriented *efficiency* point of view rather than a theoretical *feasibility* point of view. Let us illustrate the phenomenon with two examples:

Example 1. Consider the language $\mathcal{L} = \{\langle n, g, h, y\rangle \mid \exists s, t : y = g^s h^t \bmod n\} \subseteq \mathcal{L}_{\mathsf{in}} = \mathbb{N}_k^4$ where $\mathbb{N}_k$ is all k-bit numbers and the following variation of the standard Schnorr proof: the prover sends the value $u = g^{s_0} h^{t_0}$ for some random integers s_0, t_0; upon receiving u the verifier responds with some integer c and finally the prover responds with $s_1 = s_0 - c \cdot s$ and $t_1 = t_0 - c \cdot t$ (calculated over the integers). The verifier returns 1 if and only if $u = y^c g^{s_1} h^{t_1} \bmod n$. This protocol has been used numerous times (see e.g., [23,15,1]). However the protocol *is not a proof of knowledge*: on the one hand, in the case that the factorization of n is easy, it is feasible to design a knowledge extractor that in expected polynomial time can recover the witness to the statement when interacting with any convincing prover. Nevertheless such extractor can only succeed for certain choices of y as the above protocol can make the verifier accept with high probability even for "malformed" y's that satisfy $y = \zeta g^s h^t$ where ζ

is a small order element of $\mathbb{Z}_n^*$. Furthermore, when the factorization of n is difficult, the knowledge extractor cannot even take advantage of Chinese remaindering to process the values submitted by the prover; in such case ensuring the verifier that a convincing prover is indeed in possession of a witness becomes even more elusive. In addition, observe that the zero-knowledge property is affected by the way the protocol is executed, and in particular the statistical zero-knowledge aspect of the above protocol depends on the relative sizes of s_0, s and t_0, t.

Example 2. Consider the language $\mathcal{L} = \{\langle n, g, y\rangle \mid \exists s, r : y = g^{s^2} h^r\}$. A way for designing an efficient protocol for this language is to have the prover provide a commitment $C = g^s h^{r'}$ and then prove simultaneously the knowledge of the commitment C as well as the commitment C^s using two instances of the protocol in example 1. Clearly, in this case we will have to deal with similar issues as in example 1, but furthermore we will have an additional difficulty to simulate the value C as part of the zero-knowledge simulator. For choices of the values of g, h, n where $\langle h\rangle$ happens to be a subgroup of $\mathbb{Z}_n^*$ different than $\langle g\rangle$ it can be the case that C is not sufficiently hiding its g^s component. For example $\langle h\rangle$ can be the subgroup of quadratic residues in $\mathbb{Z}_n^*$ and g a quadratic non-residue; this choice would be leaking one bit about the committed value s.

The above two cases exemplify the fact that there are many efficient protocols that are not zero-knowledge proofs but they may potentially be used as such as long as they are employed over a suitable input generation. It follows that given the state of the art what is badly missing is a *methodological,* i.e, a formal way to guide cryptographic protocol designers under what conditions (on input and verifier's state) it is safe to deploy these efficient protocols as subroutines in a larger application context. Identifying such safety conditions and attaching them to a protocol is what we call "identifying the protocol's *portability*."

We say that a protocol is *portable* with safety conditions defined by a class of input generators, for the class over which it retains the properties of zero-knowledge proof of knowledge. The lack of properly identifying this notion has created a number of crucial protocol problems on previously published works. For example, the work of [23] has been cited extensively and its results were used directly to justify the proof of knowledge properties of various proposed schemes. This was done without realizing that some of the security arguments in [23] are incorrect, which was finally noticed (and corrected but without providing a formal protocol framework) by Damgård and Fujisaki [21] five years after the publication of the original paper. Further, in various cases the possibility of a biased input generation and reference string contribution by one of the parties was not considered (either in the model or as an omission or as an oversight) and this led to other works pointing out actual problems in these cases. For example, see the attack of [16] on [1] that illustrates how a malicious key generation leads to a soundness attack in the underlying signing protocol that, in turn, enables a framing attack in the group signature scheme. Another example is the attack of [29] on [5] that takes advantage of a malicious parameter generation to break the zero-knowledge property of the protocol construction. In both cases the required properties can be preserved by ensuring proper parameter generation (as it was argued in [2] and [5] respectively). These previous problem instances highlight the need of having a proper formalism that identifies conditions for porting efficient protocols as zero-knowledge proofs.

Our Contributions.

1. We introduce the notion of *portability* for proofs of knowledge protocols which identifies input and initial constraints under which a protocol can be employed and have the zero-knowledge proof properties. First, we define the notion of an *input-generator* for a proof protocol and we formalize the properties of soundness and zero-knowledge conditional on a given input generator. The portability of the protocol is defined, in turn, by identifying classes of input generators for which the protocol is sound and zero-knowledge (thus, can be deployed safely). Note that *unconditional portability* characterizes protocols that retain their properties for any input distribution (i.e., this notion coincides with regular zero-knowledge proofs of knowledge).
2. We then identify a large class of input generation and soundness parameters over which Generalized Schnorr Proofs (GSP) are portable. This clarifies the correct way to employ the highly popular protocol description notation introduced in [14,15] for GSP mentioned above. Based on our results the (frequently lacking and often erroneous) security analysis of all these previous works is streamlined and presented in a unified way. Indeed, the notation $\mathsf{PK}(\alpha, \ldots : y = g^{\alpha}, \ldots)$ was originally suggested for a few specific protocols without clear semantics and syntax for the notation nor with a way to derive a concrete protocol for the notation. Subsequently, the notation was extended by many authors and was also used in different (algebraic) settings thereby opening gaps between statement made in the notation and the security properties offered by the protocol that the authors seemingly had in mind. Sometimes, the notation has also been used with no particular protocol in mind but just to describe any protocol (e.g., a generic zero-knowledge proof protocol) that proves knowledge of a witness to the statement. This leads to our next contribution.
3. We introduce a new notation PKspec for specifying GSP proofs that puts forth the soundness guarantees provided by the protocol specified by it. Our notation can be used as a black-box in protocol design and the respective security proofs. To illustrate our notation, as an example, consider two parties that jointly compute the values U, V, n such that $U, V \in \mathbb{Z}_n^*$ and one of them wishes to demonstrate a certain structural relationship between them. This goal will be specified syntactically in the following way (for example):

$$\mathsf{PKspec}(\alpha_1, \alpha_2 : (V = h^{\alpha_1} g^{\alpha_2} \text{ in } \mathbb{Z}_n^*) \wedge \alpha_1 \in [-\infty \ldots + \infty] \wedge \alpha_2 \in [L \ldots R])$$
$$\rightarrow (\alpha_1, \alpha_2 : (V = \zeta \cdot h^{\alpha_1} g^{\alpha_2} \text{ in } \mathbb{Z}_n^*) \wedge \alpha_1 \in [-\infty \ldots + \infty] \wedge \alpha_2 \in [L' \ldots R'])$$

Note that the specification is divided into two parts, the one appearing in the first line is what the protocol designer (ideally) wishes to ensure and the second is what will actually be ensured by the Schnorr protocol (in particular, the values ζ_1, ζ_2 will be selected from some small subgroup and the range $[L', R']$ may be extended compared to $[L, R]$). Based on our work, a protocol designer may write a GSP specification as above and then rely on our analysis for the proof of a security and soundness (which assures portability of the GSP protocol to his/ her specific context).

4. To complete the tool kit for protocol designers, we introduce an efficient extension of GSP protocols that is unconditionally portable. This construction is proven correct and secure in the standard model, whereas the only previously known efficient protocols — known as the class of Σ^+ protocols [5] — were shown secure in the random oracle idealization.
5. The identification of portability for Generalized Schnorr Proofs facilitates the correct and secure design of efficient protocols. To illustrate the power of our framework in this context we consider two well-known cryptographic constructions from different subareas. We show how the employment of our GSP framework clarifies their design and the assumptions they depend on, and assures their security while coping with previously presented attacks. We first consider the original scalable group signature scheme by Ateniese et al. [1] mentioned earlier. Recently, [16] presented an attack (which is actually based on considering the extended setting of dishonest group manager at the system's setup phase, something not originally anticipated; see [2] for a discussion). Employing the GSP framework, in turn, allows us to clarify the settings where the protocol of [1] is secure and highlights the exact requirements on the joint input to the proof of knowledge. As a side benefit our framework also shows how the scheme can be made more efficient. Next, we consider the efficient divisible e-cash scheme of Chan et al. [17]; the security of this scheme was never analyzed properly (and originally the scheme as published had problems). Employing our GSP framework here, we reveal the exact cryptographic assumptions required for the modified scheme to be secure (something that even the corrected version [18] has been lacking).

   Due to lack of space the above contributions are included in the full version of the paper available in [12].

How to use the results of this paper in cryptographic protocol design. Here we comment briefly on the way our results can be used in cryptographic design. Suppose that in a certain cryptographic system a party is required to execute a proof that involves a series of discrete-log relations expressed in the widely used ad-hoc PK notation. Using Theorem 1 the designer can obtain the corresponding PKspec expression and, by the same theorem also automatically get the GSP protocol implementing the proof. Then the designer examines the input generation that preceeds the protocol which is defined by the system execution until the moment the GSP protocol should be invoked; if the conditions of Theorem 1 are satisfied then the soundness and the zero-knowledge property are implied immediately. If on the other hand, the conditions of Theorem 1 are not met, then the designer may use the unconditionally portable transformations of GSP protocols presented in section 6. For two concrete examples the reader can refer to the full version of the paper [12].

2 Preliminaries

Notations. A function $f : \mathbb{N} \to \mathbb{R}$ is called negligible if for all $c \in \mathbb{R}$ there exists $\nu_0 \in \mathbb{N}$ so that for all $\nu \geq \nu_0$ it holds that $f(\nu) < \nu^{-c}$. When a random variable x is distributed according to the probability distribution X with support S we will write

$\mathbf{Prob}_{x \leftarrow X}[x = s]$ for the probability that x takes the value $s \in S$. Let x, y be two random variables with the same support $S(\nu)$ distributed according to the probability distributions $X(\nu), Y(\nu)$ where $\nu \in \mathbb{N}$. We say that x, y are statistically indistinguishable if the function $f(\nu) := \frac{1}{2} \sum_{s \in S(\nu)} |\mathbf{Prob}_{x \leftarrow X(\nu)}[x = s] - \mathbf{Prob}_{y \leftarrow Y(\nu)}[y = s]|$ is a negligible function. If $m \in \mathbb{N}$ we will use the notation $[m]$ to denote the set $\{0, \ldots, m-1\}$. In general we will denote by $\mathcal{L}$ some language typically over alphabet $\{0,1\}$ unless otherwise specified. If $\mathcal{L}$ is an NP language, $R_{\mathcal{L}}$ will be the corresponding polynomial-time relation, i.e., $\mathcal{L} = \{\phi \mid \exists w : (\phi, w) \in R_{\mathcal{L}}\}$.

Interactive Protocols. Let $\Pi = (P, V)$ be a protocol where P, V are probabilistic interactive Turing machines (ITM). The *view* of P in Π is a random variable that contains all messages exchanged with V as well as the contents of all tapes of P. Two protocols $\Pi_1 = (P_1, V_1), \Pi_2 = (P_2, V_2)$ can be concatenated if we execute first (P_1, V_1) and then write the private outputs of P_1, V_1 to the input tapes of P_2, V_2 respectively and start the execution of (P_2, V_2). We allow parties to output a special symbol $\perp$ to signify that they "reject" a certain interaction. In the context of sequentially composed protocols, producing a $\perp$ symbol at some intermediate stage would signify that a party refuses to continue with the execution (and the final output of the party becomes $\perp$ which may interpreted as reject in the context of zero-knowledge proofs). For a given protocol $\Pi = (P, V)$ we will say that the two ITM's V, V' are indistinguishable provided that in the context of the Π interaction it is impossible for any adversarial P to distinguish whether it is communicating with V or V' (the notion is defined similarly for the case of the ITM's P, P').

3 Portability of Zero-Knowledge Proofs

A zero-knowledge proof protocol $\Sigma = (P, V)$ for a language $\mathcal{L}$ enables P to demonstrate to V that a joint input t belongs to an NP language $\mathcal{L}$ provided that the prover possesses a witness w such that $(t, w) \in \mathcal{R}_{\mathcal{L}}$. Soundness and zero-knowledge of such protocols should hold for any input distribution. Here we consider the (non-limiting) case that the prover and the verifier collaboratively construct the input t to the proof protocol by engaging in a protocol Π (dubbed the "input-generator"); at this preamble stage we denote the two parties by $P_{\mathsf{in}}, V_{\mathsf{in}}$ to highlight their relation with the actual prover and verifier. The output of this preamble stage will be the input to the actual prover and verifier.

Definition 1. *Let $\mathcal{L}_{\mathsf{in}} \in \mathsf{BPP}, \mathcal{L} \in \mathsf{NP}$ with $\mathcal{L} \subseteq \mathcal{L}_{\mathsf{in}}$. Consider Π, a two-party protocol $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ where each party may reject returning $\perp$ while if P_{in} terminates successfully it returns a pair $\langle t, w_P \rangle$ and similarly V_{in} returns $\langle t', w_V \rangle$ where $t, t' \in \mathcal{L}_{\mathsf{in}}$. The protocol Π is called an* input generator *for $\mathcal{L}$, if for all executions that neither party returns $\perp$ it holds that $(t, w_P) \in R_{\mathcal{L}}$ and $t = t'$.*

Next we define statistical zero-knowledge proofs of knowledge over input generators. The definition follows the standard ZK notion with the only difference being that the input instead of being totally adversarial (i.e., universally quantified) is produced by an input generator protocol Π. The parties *are allowed* to be adversarial during this input

generation stage. In particular for soundness we allow the prover to bias the input generation and in formalizing soundness the knowledge extractor will be interacting with the malicious prover in both stages (with rewinding power only during the second stage, i.e., the proof system). Regarding zero-knowledge we condition on all input generation executions that the honest prover agrees to execute the proof system and we require the existence of a simulator that can simulate the view of any malicious verifier. Note further that to support design flexibility we will allow the prover to show that the input belongs to a possibly extended language $\mathcal{L}_{\mathsf{ext}}$.

Definition 2. *The two party protocol $\Sigma = \langle P, V \rangle$ is a zero-knowledge proof of knowledge over the input generator $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ for $\mathcal{L}$ with knowledge error parameters $(\mathcal{L}_{\mathsf{ext}}, \kappa)$ and zero-knowledge distance ϵ if these properties are satisfied:*
(1) Completeness: it holds that both P_{in} and V_{in} terminate successfully with overwhelming probability and subsequently V accepts the interaction with the prover P with overwhelming probability.
(2) Soundness: For any pair of (P^*_{in}, P^*) we denote by $\pi_{P^*_{\mathsf{in}}, P^*}$ the probability that P^* convinces V on inputs generated by P^*_{in} and V_{in} (where $\pi_{P^*_{\mathsf{in}}, P^*}$ is taken over the entire probability space of $(P^*_{\mathsf{in}}, V_{\mathsf{in}}), (P^*, V)$). We say that Σ is sound over Π, if there is some K_{in}, such that: (i) K_{in} and V_{in} are indistinguishable as ITM's, (ii) for any P^* there is some K for which it holds that for any P^*_{in}: K on input the view of K_{in} and the output of P^*_{in}, it returns w' such that $(\mathsf{t}, w') \in R_{\mathcal{L}^{\mathsf{ext}}}$ where t is the statement that is determined in the input generation stage between P^*_{in} and K_{in} with probability of success at least $\mathsf{c} \cdot \pi_{P^*_{\mathsf{in}}, P^*}$ where $\mathsf{c} \in \mathbb{R}$ while running in time polynomial in $(\pi_{P^*_{\mathsf{in}}, P^*} - \kappa)^{-1}$.
(3) Zero-knowledge: Σ is statistical ZK over Π, if there exists an S_{in}, such that (i) S_{in} and P_{in} are indistinguishable as ITMs, (ii) for any V^*, there is a simulator S, such that for any V^*_{in}: the random variable that equals the view of V^* when interacting with P on input generated by $P_{\mathsf{in}}, V^*_{\mathsf{in}}$ is distinguishable with distance at most ϵ from the random variable that equals the output of S given as input the view of S_{in} and the output of V^*_{in}.

We next introduce the notion of portability of a protocol:

Definition 3. *The two party protocol $\Sigma = \langle P, V \rangle$ is said to be* portable *over the class of input generators $\mathcal{W}$ if for all $\Pi \in \mathcal{W}$ it holds that Σ a zero-knowledge proof of knowledge over Π. If $\mathcal{W}$ contains all possible protocols then the protocol Σ is said to be* unconditionally portable.

Ensuring portability from semi-honest behavior. Suppose that a given protocol happens to be a zero-knowledge proof of knowledge for some input-generator Π as long as the prover and the verifier are semi-honest at the input generation stage. In such an occasion one can generically compile a protocol Σ^* from Π and Σ so that Σ^* becomes a zero-knowledge proof of knowledge over Π using the transformation from semi-honest to malicious behavior put forth in [26] (see also [27], section 7.4). Note that while this is feasible, it is not particularly efficient given that it requires expensive steps such as coin-flipping combined with generic zero-knowledge proofs to ensure that no party is deviating from the input distribution (recall that much of cryptographic protocol design is motivated by avoiding generic inefficient tools). Our results will demonstrate

that such generic techniques can be substituted by much more efficient ones for the particular class of protocols we consider (i.e., generalized Schnorr proofs).

Comparison to common-reference-string/bare model ZK. Zero-knowledge proofs are sometimes modeled in the common-reference string model, cf. [20] (or the common random string model, [36]); in this setting there is an explicit separation between the input of parties and the reference string that is assumed to be honestly generated and provided to the parties. A common-reference-string ZK protocol is supposed to satisfy the security properties conditional on the distribution of the reference string that no party can bias. By comparison, in our setting there is no unbiased reference string that is independent of the proof's statement that can be used to assist in the proof of soundness or zero-knowledge. While here we deal mainly with the bare model, it is worth noting that even the availability of a common reference string does not eliminate the issues of context dependent contributed inputs.

Relaxed Knowledge Extraction. In our formulation, the knowledge extractor only ensures that the prover possesses knowledge of a witness showing that t belongs to an extended language $\mathcal{L}_{\mathsf{ext}}$. If $\mathcal{L} = \mathcal{L}_{\mathsf{ext}}$ the soundness definition will ensure that the interactive input belongs to $\mathcal{L}$ (as in the standard definition of ZK), however we will also consider slightly different languages $\mathcal{L}_{\mathsf{ext}}$. The reason for this relaxation is that by extending the language one may obtain more efficient protocols which is our primary concern. Naturally this will allow the prover to convince the verifier to accept despite the fact that the interactive input may be in the "gray area" $\mathcal{L}_{\mathsf{ext}} - \mathcal{L}$. Note that in principle we will always be able to modify the interactive input proof of knowledge so that $\mathcal{L} = \mathcal{L}_{\mathsf{ext}}$ (if one does not mind the additional computation overhead that will be incurred).

Sigma Protocols. Our applications will focus on protocols $\langle P, V \rangle$ that are called Σ-protocols, i.e., a three-move protocol in which the prover goes first, the verifier responds with a random challenge from $\{0,1\}^k$, the prover responds, and finally the verifier either accepts or rejects based on the prover's response. All conversations in a Σ-protocol are of the form $\langle com, c, res \rangle$ (commitment, challenge, response). These protocols typically consider the setting where the verifier is restricted to be "honest" during the interactive proof $\langle P, V \rangle$ when proving the zero-knowledge property. While we will follow this, however, we will still allow the verifier to be totally adversarial in the input building stage. This is justified as the honest verifier setting can be transformed using numerous techniques to the fully adversarial verifier setting (e.g. see [35,20]) and these techniques readily apply to our setting.

Variations of the definition. In our definition we focused on knowledge extraction following the definition of [6] (note that in our protocols the knowledge error will be $\kappa = 2^{-k}$ where k is a parameter). Moreover we formulated zero-knowledge in the statistical sense. It is easy to reformulate the definition by strengthening zero-knowledge (e.g., perfect zk) or relaxing it (e.g., computational zk). Moreover, soundness can be relaxed to require only language membership from the prover (instead of knowledge extraction), or defined with a specialized knowledge extractor that extracts two accepting conversations with the same first move and then reconstructs the witness. Further, in applications the protocols can be made non-interactive employing the Fiat-Shamir heuristics [22] and then use the forking Lemma [35] for extraction in the random

oracle model. These alternative definitions are well understood in the context of building efficient zero-knowledge proofs and can be ported into our setting.

On the input generation stage. In an actual system, the input generator protocol $\langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ may abstract many parties and involve interactions between many participants. From a ZK security point of view, P_{in} will comprise the "prover side" (i.e., the side that is interested in preserving zero-knowledge) and V_{in} will comprise the "verifier side" (i.e., the side of the system that is interested in in preserving soundness). In a multi-party system, we will be interested in primarily two input generators: in the first one, P_{in} will include only the prover and (if it exists) any party the prover trusts while V_{in} will include all other participants. In the second one, V_{in} will include the verifier and (if it exists) any party the verifier trusts, while P_{in} will include all other participants. If a protocol is portable over both of these input generators then it can be safely deployed in the given system.

A central tool in our design is the notion of safeguard groups that we introduce next.

4 Safeguard Groups

A safeguard group is specified by a sampler algorithm S_{sg} that on input 1^ν returns a tuple $\langle \mathbb{G}, g, M, k, \zeta \rangle$; where $\mathbb{G}$ is a description of an Abelian group that contains an implementation of $\mathbb{G}$'s binary operator, inverse computation, the encoding of 1 as well as the description of a polynomial-time group membership test that, given any string, it decides whether it is a proper encoding of a group element; g is a generator of $\mathbb{G}$; M is an approximation of the order of g in $\mathbb{G}$; and k is a security parameter that is related to the length of the order of small-order group elements. Note that we will use the same notation for the description of a group $\mathbb{G}$ and the group itself. Regarding the remaining elements of the tuple we have that $g \in \mathbb{G}, \zeta \subseteq \mathbb{G}, M \in \mathbb{N}$ with further properties to be specified below.

Definition 4. *A safeguard group sampler S_{sg} satisfies the following (where $\langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$):*

C1. *The exponent of $\mathbb{G}$ is not divisible by the square of any k-bit integer.*
C2. *The order m of g in $\mathbb{G}$ has no k-bit integer divisor, and M satisfies that $(M - m)/M = \mathsf{negl}(\nu)$.*
C3. *ζ contains only a polynomial (in ν) number of elements; they all have a known (say part of the subgroup description) k-bit integer order.*
C4. **Small-Order Property.** *It is hard to find k-bit order elements of $\mathbb{G}$ outside ζ. Formally, it holds that for all PPT $\mathcal{A}$, $\mathbf{Prob}[(v \notin \zeta) \wedge (v \text{ has } k \text{ bit order}); v \leftarrow \mathcal{A}(1^\nu, \tau); \tau = (\mathbb{G}, g, M, k, \zeta) \leftarrow S_{\mathsf{sg}}(1^\nu)] = \mathsf{negl}(\nu)$.*
C5. **Strong-Root Property.** *Given $z \in \langle g \rangle$ it is hard to find $e > 1$ and $u \in \mathbb{G}$ such that $u^e = z$. Formally, it holds that for all PPT $\mathcal{A}$, $\mathbf{Prob}[(u^e = z) \wedge (e > 1); \langle u, e \rangle \leftarrow \mathcal{A}(1^\nu, \tau, z); z \leftarrow_R \langle g \rangle; \tau = (\mathbb{G}, g, M, k, \zeta) \leftarrow S_{\mathsf{sg}}(1^\nu)] = \mathsf{negl}(\nu)$.*

We remark that properties C3-C4 are not really essential and can be dropped at the expense of loosing tightness in some of our proof reductions and notational presentation; we opt to enforce them as they make the presentation of the results more succinct and are easily satisfied for the known examples of safeguard groups.

Example 1. A safeguard group distribution can be built as follows: sample n as a safe composite so that $n = pq$, $p = 2p' + 1, q = 2q' + 1$, where p', q' are prime numbers larger than 2^k, set $\mathbb{G} = \mathbb{Z}_n^*$ and let g be a generator of quadratic residues modulo n. Finally set $\zeta = \{1, -1\}$ and $M = \lfloor \frac{n}{4} \rfloor$. Property C1 is immediate as the exponent of $\mathbb{Z}_n^*$ is $2p'q'$. Observe also that the properties C2 and C3 are easily satisfied. Indeed, it is easy to see that M is sufficiently close to $p'q'$. Next observe that a violation of property C4 would mean the recovery of any other element that has a k-bit order outside $\{1, -1\}$; this would violate the factoring assumption (only the four square roots of 1 are k-bit order elements in $\mathbb{Z}_n^*$ based on our selection of n). Property C5 amounts to the Strong-RSA assumption with the target challenge being an arbitrary element of the quadratic residues; this is a variant of the strong RSA problem that has been utilized extensively in previous works (e.g., [19]).

Example 2. A second safeguard group is over the group $\mathbb{G} = \mathbb{Z}_{n^2}^*$ where n is sampled as before, i.e., $n = pq$, $p = 2p' + 1, q = 2q' + 1$, so that g is a generator of the subgroup of square n-th residues; as before we select p', q' larger than 2^k and $\zeta = \{1, -1\}$.

We remark that in both the above examples it is not necessary to select n as a safe composite, i.e., we may allow p' and q' to be composite numbers themselves as long as they have no small divisors (of k-bits). In practical settings where we will employ safeguard groups, the parameter k may be required to be in the range from 80 to 256 bits.

Properties of Safeguard Groups. In the first lemma below regarding safeguard groups we show that based on the properties of the safeguard group it is hard for an adversary to produce arbitrary powers of a chosen power of a group element. This lemma is an important building block of our general proof protocol. We remark that various restricted special case incarnations of this lemma have appeared in the literature (the most basic of which is referred to as Shamir's trick and corresponds to case (i) in the proof of lemma). These special incarnations are too restricted to be useful in our setting and thus there is need for putting forth the lemma that is formulated as follows:

Lemma 1. *Let $\tau = \langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$ be a safeguard group distribution. Suppose that $\mathcal{A}$ is a PPT that given τ and a random $z \in \langle g \rangle$ returns $y \in \mathbb{G}$ and $t, m \in \mathbb{Z}$ such that $y^t = z^m$ with $1 \leq \gcd(t, m) < |t|$ and t is a k-bit integer. It holds that the success probability of $\mathcal{A}$ is negligible in ν.*

Our main result regarding safeguard groups is Lemma 3. We show that any adversary that is given any number of bases from the $\langle g \rangle$ subgroup of the safeguard group is incapable of producing an entirely arbitrary discrete-log representation of a power of his choosing within $\mathbb{G}$. Before stating the main lemma, we show an auxiliary lemma.

Lemma 2. *Let A, B be two integers with $A > B$ and $A = \pi B + v$ with $0 \leq v < B$ and let X be a random variable with $X \leftarrow_R [A]$. Let $Y = X \bmod B$. The statistical distance of the distribution of Y and the uniform distribution over $\mathbb{Z}_B$ is at most v/A. Let $Y' = \lfloor X/B \rfloor$. The statistical distance of the uniform distribution over $\{0, \ldots, \pi\}$ and the distribution of Y' is at most $1/(\pi + 1)$.*

Lemma 3. *Let $B_1, \ldots, B_r \leftarrow_R \langle g \rangle$, $\langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$ be a safeguard group distribution, and let $\mathcal{A}$ be a* PPT *that on input $\mathbb{G}, g, M, k, \zeta, B_1, \ldots, B_r$ it outputs integers $e_1, \ldots, e_r, t$ and $y \in \mathbb{G}$ such that with probability α: $|t| > 1$ and $\prod_{i=1}^{r} B_i^{e_i} = y^t$ where t is a k-bit number and $\exists i : t \not| e_i$. Then the Strong-Root property is violated with probability at least $\alpha/(2r+1) - \eta$ where η is a function negligible in ν.*

5 The Portability of Generalized Schnorr Proofs

In this section we discuss the portability of Generalized Schnorr Proofs. In particular we will identify a wide class of input generators so that under the right conditions these protocols are portable.

GSP-specs. A generalized Schnorr proof (GSP) operates on a statement $\mathtt{t}$ that involves a number of groups and group elements ("bases") with public and secret exponents. To any such statement $\mathtt{t}$ we will associate the following:

i. A set of symbolic variables denoted by $\mathcal{X} = \{\alpha_1, \ldots, \alpha_r\}$ with $|\mathcal{X}| = r$.
ii. A sequence of group descriptions $\mathbb{G}_1, \ldots, \mathbb{G}_z$ as well as the descriptions of z subgroups $\zeta_1, \ldots, \zeta_z$ of $\mathbb{G}_1, \ldots, \mathbb{G}_z$ respectively, so that the exponent of each ζ_i is (at most) a k-bit integer. The description of the subgroup ζ_i will be typically given as a list of elements (i.e., these subgroups are small). It may be the case that $\zeta_i = \{1\}$.
iii. The group elements $A_{i,j} \in \mathbb{G}_i$ for $j = 0, \ldots, r$ where $A_{i,j}$ will be the base for the variable α_j in group $\mathbb{G}_i$.
iv. The range limits $L_j, R_j, L_j^{\mathsf{ext}}, R_j^{\mathsf{ext}} \in \mathbb{Z} \cup \{-\infty, \infty\}$ such that $L_j < R_j$, and $L_j^{\mathsf{ext}} \leq L_j, R_j \leq R_j^{\mathsf{ext}}$ for $j = 1, \ldots, r$.

Next we give an explicit syntax notation and semantics for specifying the language $\mathcal{L}$ that the prover wishes to convince the verifier the statement $\mathtt{t}$ belongs to. We define two languages $\mathcal{L}$ and $\mathcal{L}^{\mathsf{ext}}$:

$$\mathcal{L} = \Big\{ t \in \mathcal{L}_{\mathsf{in}} \mid \exists\, x_i \in \mathbb{Z} : \bigwedge_{i=1}^{z} \Big(\prod_{j=0}^{r} A_{i,j}^{x_j} = A_{i,0} \Big) \wedge \bigwedge_{j=1}^{r} \Big(x_j \in [L_j, R_j] \Big) \Big\}$$

$$\mathcal{L}_{\mathsf{ext}} = \Big\{ t \in \mathcal{L}_{\mathsf{in}} \mid \exists\, x_i \in \mathbb{Z} : \bigwedge_{i=1}^{z} \Big(\prod_{j=0}^{r} A_{i,j}^{x_j} = \zeta_i \cdot A_{i,0} \Big) \wedge \bigwedge_{j=1}^{r} \Big(x_j \in [L_j^{\mathsf{ext}}, R_j^{\mathsf{ext}}] \Big) \Big\}$$

We will use the following syntax to refer to a proof of knowledge for the language $\mathcal{L}$ whose soundness is only ensured in the extended language $\mathcal{L}_{\mathsf{ext}}$; we call this notation a GSP-spec τ.

$$\mathsf{PKspec}\Big(\mathcal{X} : \prod_{j=1}^{r} A_{1,j}^{\alpha_j} = A_{1,0}(\text{in } \mathbb{G}_1) \ldots \wedge\ \alpha_1 \in [L_1, R_1] \wedge \ldots \wedge \alpha_r \in [L_r, R_r] \Big)$$

$$\rightarrow \Big(\mathcal{X} : \prod_{j=1}^{r} A_{1,j}^{\alpha_j} = \zeta_1 \cdot A_{1,0}(\text{in } \mathbb{G}_1) \ldots \wedge \alpha_1 \in [L_1^{\mathsf{ext}}, R_1^{\mathsf{ext}}] \wedge \ldots \wedge \alpha_r \in [L_r^{\mathsf{ext}}, R_r^{\mathsf{ext}}] \Big)$$

Note that left-hand side of the above notation (i.e., the first line) is the statement of the proof whereas the right-hand side (namely, the second line) is the actual (extended) statement that will be guaranteed to hold (recall Definition 2). Note that in the extended statement the ranges $[L_j, R_j]$ will be extended to $[L_j^{\mathsf{ext}}, R_j^{\mathsf{ext}}]$ and the unit element of the group is extended to be any element in the (small) subgroup ζ_i for the i-th equation.

The specification allows for a wide range of proofs including polynomial relations among the secret and inequality statements of secrets. We refer to in the full version of the paper [12] for a discussion on what is covered by this specification and how it can be extended, in particular to include also $\vee$-connectives or tighter ranges.

GSP input generators. A GSP input generator $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ that is consistent with a GSP-spec τ is a two party protocol that determines the parameters: z (the number of groups), r (the number of symbolic variables), k (a parameter related to group selection and the soundness property) and whose public output $\mathtt{t}$ includes the description of all groups, bases and ranges of the GSP-spec as described in the items (i)-(iv) above.

The Generalized Schnorr Protocol $\Sigma_\tau^{\mathsf{GSP}}$. For any GSP-spec τ one can design a Sigma protocol based on Schnorr's proof by introducing appropriate range checking and compensating for the fact that groups of unknown order are used with computations over the integers.

The protocol is based on two parameters k, l for free variables $\alpha_1, \ldots, \alpha_r$ such that α_j takes values in the range $[L_j, R_j]$. Below we set $m_j = R_j - L_j$. Suppose the prover is in possession of the witnesses $x_1, \ldots, x_r$; the prover selects first the random values $t_j \in_R [-2^{k+l}m_j, 2^{k+l}m_j]$ and computes the values $B_i = \prod_{j=1}^{r} A_{i,j}^{t_j}$. The prover terminates the first stage of computation by transmitting $B_1, \ldots, B_z$. The verifier selects $c \in_R \{0,1\}^k$ and responds by sending c to the prover. The prover, in response computes the integers $s_j = t_j - c \cdot (x_j - L_j)$ and sends them to the verifier. The verifier returns 1 if and only if for all $j \in \{1, \ldots, r\}$ it holds that $s_j \in [-2^{k+l}m_j - (2^k - 1)m_j, 2^{k+l}m_j]$ as well as for all $i \in \{1, \ldots, z\}$ it holds that $B_i \in \mathbb{G}_i$ and $\prod_{j=1}^{r} A_{i,j}^{s_j} =_{\mathbb{G}_i} B_i (A_{i,0}^{-1} \cdot \prod_{j=1}^{r} A_{i,j}^{L_j})^c$.

Portability of $\Sigma_\tau^{\mathsf{GSP}}$. We will next identify a class of input generators Π for a given GSP-spec τ over which $\Sigma_\tau^{\mathsf{GSP}}$ is portable as a zero-knowledge proof of knowledge. Recall that Π defines the respective inputs $(\mathtt{t}, w)$ for the prover and $\mathtt{t}$ for the verifier. We first describe the setting where some special care needs to be paid when arguing the security of $\Sigma_\tau^{\mathsf{GSP}}$. These settings involve variables that are "unsafe":

Definition 5. ***(Unsafe Variables)*** *For a GSP-spec τ, a symbolic variable $\alpha_j \in \mathcal{X}$ is called unsafe if it satisfies at least one of the following three conditions: (1) it is involved in an equation over a group $\mathbb{G}_i$ over a base element that is of unknown order to the verifier (i.e., the order of the base is not included in the group's description); (2) the range $[L_j, R_j]$ is non-trivial (i.e., it is not the range $(-\infty, +\infty)$); (3) the variable appears across various bases that have known but different order.*

The presence of unsafe variables may introduce problems in the knowledge extraction argument and make the protocol fail the soundness property. Still, unsafe variables can be tolerated provided they appear in conjunction to safeguard groups (cf. Definition 4). The following definition defines input-generators that are suitable for the $\Sigma_\tau^{\mathsf{ext}}$ protocol

in the presence of unsafe variables. In a nutshell it says that for a GSP-input generator protocol Π, a certain group will be called a safeguard group *for* Π as long as there exists a simulator that playing the role of the verifier, it can "plug-in" a safeguard group generated by S_{sg} in black-box fashion in the interaction with P_{in} without P_{in} noticing, even if P_{in} is acting adversarially.

Definition 6. *For any* GSP*-input-generator protocol* $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$*, a group* $\mathbb{G}_i$ *and the bases* $A_{i,j_1}, \ldots, A_{i,j_v} \in \mathbb{G}_i$ *will be called respectively a* **safeguard group for** Π *and its* **safeguard bases** *there exists a polynomial-time simulator* S_V *s.t. for any adversarial party* P^*_{in} *in the protocol* Π*,* S_V *receives as input* $\langle \mathbb{G}, g, M, k, \zeta, g_1, \ldots, g_v \rangle$ *where* $\langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$ *and* $g_\ell = g^{s_\ell}$ *with* $s_\ell \xleftarrow{\$} [M]$*, and satisfies the property that the input* t *produced by the interaction of* P^*_{in} *and* S_V *contains a group* $\mathbb{G}_i$ *and bases* $A_{i,j_1}, \ldots, A_{i,j_v}$ *that satisfy* $\mathbb{G}_i = \mathbb{G}$ *and* $A_{i,j_1} = g_1, \ldots, A_{i,j_v} = g_v$ *and the view of* P^*_{in} *when interacting with* V_{in} *is indistinguishable from the view of* P^*_{in} *when interacting with* S_V*.*

An equation $\prod_{j=1}^{r} A_{i,j}^{\alpha_j} = A_{i,0}$ over a safeguard group for Π will be called a "safeguarding equation." Armed with the above we next identify a class of input generators for which the generalized Schnorr proof $\Sigma_\tau^{\mathsf{GSP}}$ is portable.

Theorem 1. *(Portability of Generalized Schnorr Proofs) Let* τ *be a* GSP*-spec. The protocol* $\Sigma_\tau^{\mathsf{GSP}}$ *is portable for honest verifiers, for all input generators* Π *consistent with* τ *provided that (I) the generated input* $t \in \mathcal{L}_{\mathsf{in}}$ *has no unsafe variable, or (II) the five following conditions hold: (i) Each unsafe variable appears at least once as an exponent over a safeguard base. (ii) There is an ordering* $i_1, \ldots, i_z$ *of all the equations so that (1)* i_1 *is a safeguarding equation with all its free variables over safeguard bases, and (2) in safeguarding equation* i_w *for* $w > 1$ *it holds that all free variables of equation* i_w *appear over safeguard bases or have appeared at least once in a previous safeguarding equation. (iii) If* $\mathbb{G}_i$ *is a safeguard group then it has description* $\langle \mathbb{G}_i, g_i, M_i, k, \zeta_i \rangle$ *(i.e., all safeguard groups share the same* k*). (iv)* $L_j^{\mathsf{ext}} = L_j - 2^{k+l+2}(R_j - L_j)$ *and* $R_j^{\mathsf{ext}} = R_j + 2^{k+l+2}(R_j - L_j)$*. (v) The knowledge error* κ *is* $\mathsf{c} \cdot (2^{-k} + r \cdot \mathsf{Adv}_{\mathsf{root}})$ *for a suitable* $\mathsf{c} \in \mathbb{R}$ *and the zero-knowledge distance is* $\epsilon = r \cdot 2^{-l}$*.*

Example. Suppose that V_{in} selects an RSA-modulus n which is a multiple of two safe primes, a quadratic residue base $g \in \mathbb{Z}_n^*$ as well as $h \xleftarrow{\$} \langle g \rangle$. V_{in} transmits n, g, h to P_{in}. In turn, P_{in} sends $y = g^u h^v \bmod n$ where $u \xleftarrow{\$} [\lceil \frac{n}{4} \rceil]$ and $v \in [2^e]$ for some $e \in \mathbb{N}$. The input[1] t generated by $P_{\mathsf{in}}, V_{\mathsf{in}}$ in this case is the vector $\langle n, g, h, y \rangle$. Suppose now that the prover P wishes to demonstrate to the verifier V that she knows u, v in their respective ranges such that $y = g^u h^v \bmod n$. It is easy to see that $\mathbb{Z}_n^*$ can play the role of a safeguard group for the input generator described above with $\zeta = \{-1, +1\}$ and that the conditions of Theorem 1 are satisfied, thus the protocol $\Sigma_\tau^{\mathsf{GSP}}$ can be used to ensure to V that $y = \pm g^u h^v \bmod n$ and $u \in [-E_u, \lceil \frac{n}{4} \rceil + E_u], v \in [-E_v, 2^e + E_v]$ where $E_u = 2^{k+l+2} \cdot \lceil \frac{n}{4} \rceil, E_v = 2^{k+l+2+e}$.

[1] In this simple example, it could be that y leaks some information about u, v to V_{in} (which recall it may be an entity that includes more parties beyond the verifier); this does not affect the zero-knowledge property over this input generator which — as it is the case with regular ZK proofs — is concerned only with information leaks during the P, V interaction.

6 Unconditionally Portable Protocols for GSP-specs

Theorem 1 of the previous section describes a class of input-generators for which the generalized Schnorr proof protocol can be used in a safe way. Nevertheless, it may be very well the case that we would like to use a proof for a GSP-spec outside this class of input generators. In the remaining of the section we describe an efficient protocol enhancement to the basic generalized Schnorr protocol that is unconditionally portable.

The $\Sigma_\tau^{\mathsf{ext},+}$ protocol. Consider any input generator Π for which Theorem 1 does not apply, i.e., $(\Pi, \Sigma_\tau^{\mathsf{ext}})$ is not a zero-knowledge proof over Π. We next show one modification of $\Sigma_\tau^{\mathsf{ext}}$ into a protocol $\Sigma_\tau^{\mathsf{ext}+}$ so that $\Sigma_\tau^{\mathsf{ext}+}$ is a protocol that is universally portable as a zero-knowledge proof.

The protocol $\Sigma_\tau^{\mathsf{ext}+}$ operates as follows: The verifier first selects a safeguard group $\langle \mathbb{Z}_n^*, g, M = \lfloor n/4 \rfloor, k, \mathbb{V} = \{-1, 1\} \rangle$ where $\langle g \rangle = QR(n)$ together with a number of safeguard bases $g_1, \ldots, g_u \in \langle g \rangle$ where u is the number of variables that are unsafe. We will denote the discrete-logarithm values of g_ℓ base g as ρ_ℓ. The verifier also selects a prime P such that $(P-1)/2$ is also prime and satisfies $(P-1)/2 > n$ as well as two elements of order $(P-1)/2$ in $\mathbb{Z}_P^*$ denoted by G, H where H is randomly selected from $\langle G \rangle$. When these elements are received the prover will check that $P, (P-1)/2 \in \mathsf{Prime}$, $(P-1)/2 > n$ and that $G, H \in QR(P)$ (i.e., that $H \in \langle G \rangle$). We denote $\mathsf{Adv}_{\mathtt{DLOG}}$ an upper bound on the probability that any polynomial-time bounded algorithm has in returning $\log_G(H)$ given G, H, P. Next, the prover computes a commitment of the form $C = g^r g_1^{x_1} \ldots g_u^{x_u} (\mathrm{mod} n)$ (which is an extended Pedersen commitment over the safeguard group); note that $r \stackrel{\$}{\leftarrow} [2^{l+3}M]$ where l is the security parameter related to the zero-knowledge distance and $x_1, \ldots, x_u$ are the witnesses of P. Intuitively, what will happen next can be interpreted as follows: the prover and the verifier will include in the GSP-spec τ the safeguarding equation $(C = g^r g_1^{x_1} \ldots g_u^{x_u} (\text{ in } \mathbb{Z}_n^*))$ as one of the equations that are needed to be shown (we call the extended GSP-spec τ^+) but the prover will not reveal C. This is because the parameters of the safeguard group were selected by the verifier and thus the prover is at risk of revealing some information about the witnesses.

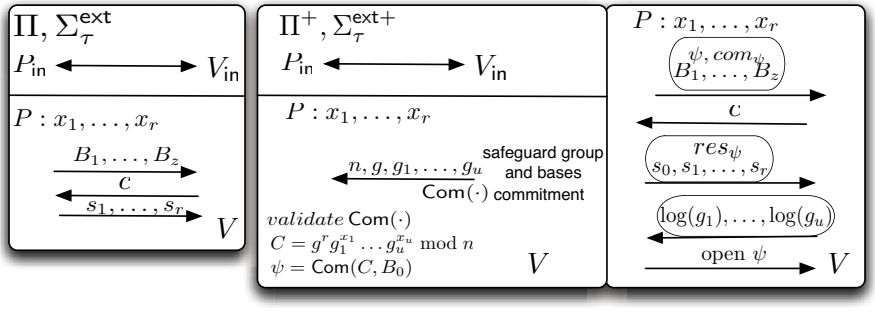


Fig. 1. Illustration of the transformation of $\Sigma_\tau^{\mathsf{ext}}$ over input generator Π to the $\Sigma_\tau^{\mathsf{ext}+}$

Instead, the (P, V) protocol interaction for τ^+ will be modified as follows: the prover P will make a commitment ψ_1 to the value C denoted by $\psi_1 = G^{r^*} H^C \bmod P$. Similarly, the prover P will not submit the value B_0 (that corresponds to the commitment equation $(C = g^r g_1^{x_1} \dots g_u^{x_u} (\text{ in } \mathbb{Z}_n^*))$); instead it will submit a commitment $\psi_2 = G^{r_0^*} H^{B_0} \bmod P$. We call $\psi = (\psi_1, \psi_2)$. Next, the prover P will need to show that ψ is well-formed; this is easy as ψ_1, ψ_2 are Pedersen commitments, so it suffices to prove knowledge of r^* and C in ψ_1 and prove knowledge of r_0^* and B_0 in ψ_2. We denote the Σ proof for the ψ commitment as com_ψ, c, res_ψ. These additional proofs can be composed in parallel AND composition with the GSP protocol $\Sigma_\tau^{\mathsf{GSP}}$ and do not incur any additional round complexity. After the verifier receives all values and accepts the proofs (except for the equation over the safeguard group), it submits to the prover the values $\rho_1, \dots, \rho_u$ who in turn checks whether $g_\ell = g^{\rho_\ell}$. In this case, the prover opens the commitments ψ_1, ψ_2, and now the verifier is able to complete the verification as described in the $\Sigma_\tau^{\mathsf{ext}}$ protocol. We illustrate the transformation in figure 1.

Remark. This transformation generalizes and improves the setting of the Σ^+ proof method introduced in [5]; it obviates the need of random oracles (their soundness argument was in the random oracle model). We note that if the number of rounds is at premium then it is possible to reduce them to 3 by giving up on other aspects of the protocol in terms of security or efficiency. Specifically, one can either have the verifier demonstrate to the prover that the safeguard group is properly selected in an "offline" stage (that will not be counting towards the rounds of the actual protocol) or assuming the existence of an auxiliary input that is honestly distributed (an approach shown in [4]).

We next prove our protocol secure for a type of "partly honest" verifiers that may operate maliciously in the safeguard group selection (i.e., the first move of the $\Sigma_\tau^{\mathsf{ext}+}$ protocol) but still select the challenge honestly (in the third move of the protocol). We choose to do this for ease of presentation as there are standard techniques that can be applied to port the protocol to the entirely malicious verifier setting (much like how an honest verifier zero-knowledge protocol can be ported to the zero-knowledge setting).

Theorem 2. *For any* GSP*-spec* τ *and any consistent input generator* Π*, the protocol* $\Sigma_\tau^{\mathsf{ext},+}$ *is an (unconditionally portable) zero-knowledge proof of knowledge over* Π *against partly honest verifiers for the same* $L_\ell^{\mathsf{ext}}, R_\ell^{\mathsf{ext}}$ *parameters as Theorem 1, knowledge error* $\kappa = \mathsf{c}(2^{-k} + \mathsf{Adv}_{\mathtt{DLOG}} + r \cdot \mathsf{Adv}_{\mathsf{root}})$ *for some* $\mathsf{c} \in \mathbb{R}$ *and zero-knowledge distance* $(r+1)2^{-l}$.

7 Demonstrative Applications and Extensions

In the full version of the paper available in [12] we provide two demonstrative applications of our framework as well as a number of possible extensions to it. The full version also includes proofs for all the statements in this version.

Acknowledgements

The authors thank Endre Bangerter for helpful discussions on the subject.

References

1. Ateniese, G., Camenisch, J.L., Joye, M., Tsudik, G.: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, p. 255. Springer, Heidelberg (2000)
2. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: Remarks on "analysis of one popular group signature scheme" in asiacrypt 2006. Cryptology ePrint Archive, Report 2006/464 (2006), http://eprint.iacr.org/
3. Ateniese, G., Song, D.X., Tsudik, G.: Quasi-efficient revocation in group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
4. Bangerter, E.: On Efficient Zero-Knowledge Proofs of Knowledge. PhD thesis, Ruhr U. Bochum (2005)
5. Bangerter, E., Camenisch, J.L., Maurer, U.M.: Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 154–171. Springer, Heidelberg (2005), http://www.zurich.ibm.com/~/jca/papers/bacama05.pdf
6. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
7. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
8. Bresson, E., Stern, J.: Efficient revocation in group signatures. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001)
9. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proc. 11th ACM Conference on Computer and Communications Security, pp. 225–234. ACM Press, New York (2004)
10. Bussard, L., Molva, R., Roudier, Y.: History-Based Signature or How to Trust Anonymous Documents. In: Jensen, C., Poslad, S., Dimitrakos, T. (eds.) iTrust 2004. LNCS, vol. 2995, pp. 78–92. Springer, Heidelberg (2004)
11. Bussard, L., Roudier, Y., Molva, R.: Untraceable secret credentials: Trust establishment with privacy. In: PerCom Workshops, pp. 122–126. IEEE Computer Society, Los Alamitos (2004)
12. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized schnorr proofs. Technical report, Cryptology ePrint Archive (2009)
13. Camenisch, J.L., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
14. Camenisch, J.L., Stadler, M.A.: Efficient Group Signature Schemes for Large Groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
15. Camenisch, J.L.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich, Diss. ETH No. 12520. Hartung Gorre Verlag, Konstanz (1998)
16. Cao, Z.: Analysis of One Popular Group Signature Scheme. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 460–466. Springer, Heidelberg (2006)
17. Chan, A.H., Frankel, Y., Tsiounis, Y.: Easy Come - Easy Go Divisible Cash. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 561–575. Springer, Heidelberg (1998)
18. Chan, A.H., Frankel, Y., Tsiounis, Y.: Easy come - easy go divisible cash. GTE Technical Report (1998), http://www.ccs.neu.edu/home/yiannis/pubs.html
19. Cramer, R., Shoup, V.: Signature schemes based on the strong rsa assumption. ACM Trans. Inf. Syst. Secur. 3(3), 161–185 (2000)

20. Damgård, I.B.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000)
21. Damgård, I.B., Fujisaki, E.: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
22. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
23. Fujisaki, E., Okamoto, T.: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
24. Furukawa, J., Yonezawa, S.: Group Signatures with Separate and Distributed Authorities. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 77–90. Springer, Heidelberg (2005)
25. Gaud, M., Traoré, J.: On the Anonymity of Fair Offline E-cash Systems. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 34–50. Springer, Heidelberg (2003)
26. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC 1987: Proceedings of the nineteenth annual ACM conference on Theory of computing, pp. 218–229. ACM Press, New York (1987)
27. Goldreich, O.: The Foundations of Cryptography, vol. 2. Cambridge University Press, Cambridge (1999)
28. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
29. Kunz-Jacques, S., Martinet, G., Poupard, G., Stern, J.: Cryptanalysis of an Efficient Proof of Knowledge of Discrete Logarithm. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 27–43. Springer, Heidelberg (2006)
30. Van Le, T., Nguyen, K.Q., Varadharajan, V.: How to Prove That a Committed Number Is Prime. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 208–218. Springer, Heidelberg (1999)
31. Lysyanskaya, A., Ramzan, Z.: Group Blind Digital Signatures: A Scalable Solution to Electronic Cash. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 184–197. Springer, Heidelberg (1998)
32. MacKenzie, P.D., Reiter, M.K.: Two-party generation of DSA signatures. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 137–154. Springer, Heidelberg (2001)
33. Mykletun, E., Narasimha, M., Tsudik, G.: Signature Bouquets: Immutability for Aggregated/Condensed Signatures. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 160–176. Springer, Heidelberg (2004)
34. Nakanishi, T., Shiota, M., Sugiyama, Y.: An Efficient Online Electronic Cash with Unlinkable Exact Payments. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 367–378. Springer, Heidelberg (2004)
35. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (2000)
36. De Santis, A., Micali, S., Persiano, G.: Noninteractive Zero-Knowledge Proof Systems. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 52–72. Springer, Heidelberg (1988)
37. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
38. Song, D.X.: Practical forward secure group signature schemes. In: Proc. 8th ACM Conference on Computer and Communications Security, pp. 225–234. ACM press, New York (2001)

39. Susilo, W., Mu, Y.: On the Security of Nominative Signatures. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 329–335. Springer, Heidelberg (2005)
40. Tang, C., Liu, Z., Wang, M.: A verifiable secret sharing scheme with statistical zero-knowledge. Cryptology ePrint Archive, Report 2003/222 (2003), http://eprint.iacr.org/
41. Tsang, P.P., Wei, V.K.: Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation. In: Deng, R.H., Bao, F., Pang, H., Zhou, J. (eds.) ISPEC 2005. LNCS, vol. 3439, pp. 48–60. Springer, Heidelberg (2005)
42. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable Linkable Threshold Ring Signatures. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 384–398. Springer, Heidelberg (2004)
43. Wei, V.K.: Tracing-by-linking group signatures. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 149–163. Springer, Heidelberg (2005)

A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks

François-Xavier Standaert[1,⋆], Tal G. Malkin[2], and Moti Yung[2,3]

[1] UCL Crypto Group, Université catholique de Louvain
[2] Dept. of Computer Science, Columbia University
[3] Google Inc.
fstandae@uclouvain.be, tal,moti@cs.columbia.edu

Abstract. The fair evaluation and comparison of side-channel attacks and countermeasures has been a long standing open question, limiting further developments in the field. Motivated by this challenge, this work makes a step in this direction and proposes a framework for the analysis of cryptographic implementations that includes a theoretical model and an application methodology. The model is based on commonly accepted hypotheses about side-channels that computations give rise to. It allows quantifying the effect of practically relevant leakage functions with a combination of information theoretic and security metrics, measuring the quality of an implementation and the strength of an adversary, respectively. From a theoretical point of view, we demonstrate formal connections between these metrics and discuss their intuitive meaning. From a practical point of view, the model implies a unified methodology for the analysis of side-channel key recovery attacks. The proposed solution allows getting rid of most of the subjective parameters that were limiting previous specialized and often ad hoc approaches in the evaluation of physically observable devices. It typically determines the extent to which basic (but practically essential) questions such as "*How to compare two implementations?*" or "*How to compare two side-channel adversaries?*" can be answered in a sound fashion.

1 Introduction

Traditionally, cryptographic algorithms provide security against an adversary who has only black box access to cryptographic devices. However, such a model does not always correspond to the realities of physical implementations. During the last decade, it has been demonstrated that targeting actual hardware rather than abstract algorithms may lead to very serious security issues. In this paper, we investigate the context of side-channel attacks, in which adversaries are enhanced with the possibility to exploit physical leakages such as power consumption [19] or electromagnetic radiation [2,14]. A large body of experimental work has been created on the subject and although numerous countermeasures are proposed in the literature, protecting implementations against such attacks

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

⋆ Associate researcher of the Belgian Fund for Scientific Research (FNRS - F.R.S.)

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 443–461, 2009.

is usually difficult and expensive. Moreover, most proposals we are aware of only increase the difficulty of performing the attacks, but do not fundamentally prevent them. Eventually, due to the device-specific nature of side-channel attacks, the comparison of their efficiency and the evaluation of leaking implementations are challenging issues, *e.g.* as mentioned in [22], page 163.

Following this state-of-the art, our work is mainly motivated by the need of having sound tools (*i.e.* a middle-ware between the abstract models and the concrete devices) to evaluate and compare different implementations and adversaries. As a matter of fact, the evaluation criteria in physically observable cryptography should be unified in the sense that they should be adequate and have the same meaning for analyzing any type of implementation or adversary. This is in clear contrast with the combination of ad hoc solutions relying on specific ideas designers have in mind. For example, present techniques for the analysis of side-channel attacks typically allow the statement of claims such as: "*An implementation X is better than an implementation Y against an adversary A*". But such claims are of limited interest since an unsuccessful attack may theoretically be due both to the quality of the target device or to the ineffectiveness of the adversary. The results in this paper aim to discuss the extent to which more meaningful (adversary independent) statements can be claimed such as: "*An implementation X is better than an implementation Y*". Similarly, when comparing different adversaries, present solutions for the analysis of side-channel attacks typically allow the statement of claims such as: "*An adversary A successfully recovers one key byte of an implementation X after the observation of q measurement queries.*". But in practice, recovering a small set of key candidates including the correct one after a low number of measurement queries may be more critical for the security of an actual system than recovering the key itself after a high number of measurement queries (*e.g.* further isolating a key from a list can employ classical cryptanalysis techniques exploiting black box queries). The results in this paper aim at providing tools that help claiming more flexible statements and can capture various adversarial strategies.

Quite naturally, the previous goals imply the need of a sound model for the analysis of side-channel attacks. But perhaps surprisingly (and to the best of our knowledge), there have been only a few attempts to provably address physical security issues. A significant example is the work of Micali and Reyzin who initiated an analysis of side-channels taking the modularity of physically observable computations into account. The resulting model in [24] is very general, capturing almost any conceivable form of physical leakage. However and as observed by the authors themselves, this generality implies that the obtained positive results (*i.e.* leading to useful constructions) are quite restricted in nature and it is not clear how they apply to practice. This is especially true for primitives such as modern block ciphers for which even the black box security cannot be proven.

In the present work, we consequently give up a part of this generality and concentrate on current attacks (*i.e.* key recovery) and adversaries (*i.e.* statistical procedures to efficiently discriminate the key), trying to keep a sound and systematic approach aside these points. For this purpose, we first separate the implementation

issue (*i.e.* "*how good is my implementation?*") and the adversarial issue (*i.e.* "*how strong is my adversary?*") in the physically observable setting. We believe that the methodological division of both concerns brings essential insights and avoids previous confusions in the analysis of side-channel attacks. As a consequence, we introduce two different types of evaluation metrics. First, an information theoretic metric is used to measure the amount of information that is provided by a given implementation. Second, an actual security metric is used to measure how this information can be turned into a successful attack. We propose candidates for these metrics and show that they allow comparing different implementations and adversaries. We also demonstrate important connections between them in the practically meaningful context of Gaussian leakage distributions and discuss their intuitive meaning. Eventually, we move from formal definitions to practice-oriented definitions in order to introduce a unified evaluation methodology for side-channel key recovery attacks. We also provide an exemplary application of the model and discuss its limitations.

Related works include a large literature on side-channel issues, ranging from attacks to countermeasures and including statistical analysis concerns. The side-channel lounge [12], DPA book [22] and CHES workshops [8] provide a good list of references, a state-of-the art view of the field and some recent developments, respectively. Most of these previous results can be re-visited in the following framework in order to improve their understanding. The goal of this paper is therefore to facilitate the interface between theoretical and practical aspects in physically observable cryptography. We mention that in parallel to our work, the models in [3,20] consider a restricted context of noiseless leakages. They allow deriving formal bounds on the efficiency of certain attacks but are not aimed to analyze actual devices (that always have to deal with noise) which is our main goal. Finally, [25] initiated a study of forward secure cryptographic constructions with rigorous security analysis of side-channel attacks. [11,26] then proposed similar constructions in a more general setting and standard model. These works exploit assumptions such as bounded adversaries or leakages of which the validity can be measured for different devices thanks to the methodology in this paper.

Finally, our analysis employs ideas from the classical communication theory [10,28,29]. But whereas source and channel coding attempt to represent the information in an efficient format for transmission, cryptographic engineers have the opposite goal to make their circuit's internal configurations unintelligible to the outside world. This analogy provides a rationale for our metrics. Note that different measures of uncertainty have frequently been used in the cryptographic literature to quantify the effectiveness of various attacks, *e.g.* in [6]. Our line of research follows a slightly different approach in the sense that we assign specific tasks to different metrics. Namely, we suggest to evaluate implementations with an information theoretic metric (conditional entropy) and to evaluate attacks and adversaries with security metrics (success rates or guessing entropy). This allows us to consider first implementations as non-adversarial information emitting objects where keys are randomly chosen, and then adversaries which operate under certain (computational and access) restrictions on top of the implementations. This

duality enables our model to be reflective of the situation in the real world and therefore to be useful beyond theoretical analysis, *i.e.* applicable to any simulated or actual lab data, for various cryptographic algorithms.

Note that because of place constraints, proofs and technical details have been removed from the paper and made available in an extended version [30].

2 Intuitive Description of the Model and Terminology

In this section, we give an intuitive description of side-channel key recovery attacks that will be formally defined and investigated in the rest of this paper.

A generic side-channel key recovery is illustrated in Figure 1 that we explain as follows. First, the term *primitive* is used to denote cryptographic routines corresponding to the practical instantiation of some idealized functions required to solve cryptographic problems. For example, the AES Rijndael is a cryptographic primitive. Second, the term *device* is used to denote the physical realization of a cryptographic primitive. For example, a smart card running the AES Rijndael can be the target device of a side-channel attack. A *side-channel* is an unintended communication channel that leaks some information from a device through a physical media. For example, the power consumption or the electromagnetic radiation of a target device can be used as side-channels. The output of a side-channel is a *physical observable*. Then, the *leakage function* is an abstraction that models all the specificities of the side-channel and the measurement setup used to monitor the physical observables. An *implementation* is the combination of a cryptographic device and a leakage function. Finally, a *side-channel adversary* is an algorithm (or a set of algorithms) that can query the implementation to get the leakage function results in addition to the traditional black-box access. Its goal is to defeat a given security notion (*e.g.* key recovery) within certain computational bounds and capabilities. Note that leakage functions and cryptographic implementations (aka physical computers) are formally defined in [24] and this paper relies on the same assumption as theirs.

Figure 1 suggests that, similarly to the classical communication theory, two aspects have to be considered (and quantified) in physically observable cryptography. First, actual implementations leak information, independently of the adversary exploiting it. The goal of our information theoretic metric is to measure the side-channel leakages in order to give a sound answer to the question: "*how to compare different implementations?*". Second, an adversary analogous to a specific decoder exploits these leakages. The goal of our security metrics is to measure the extent to which this exploitation efficiently turns the information available into a key recovery. Security metrics are the counterpart of the Bit-Error-Rate in communication problems and aim to answer the question: "*how to compare different adversaries?*". Interestingly, the figure highlights the difference between an actual adversary (of which the goal is simply to recover some secret data) and an evaluator (of which the goal is to analyze and understand the physical leakages). For example, comparing different implementations with an information theoretic metric is only of interest for an evaluator.

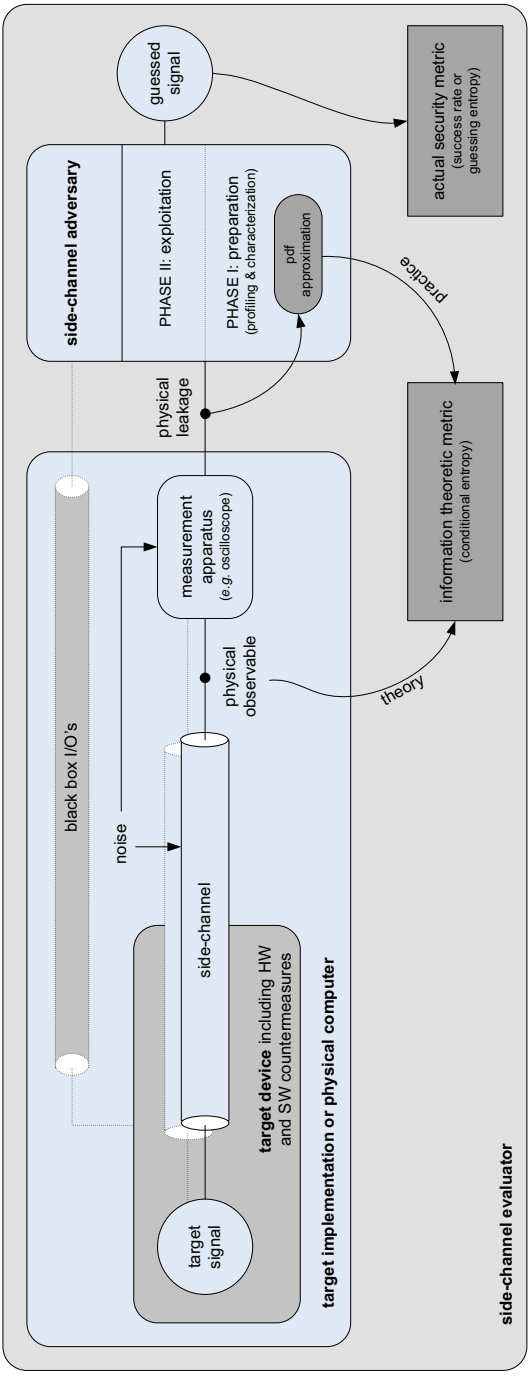


Fig. 1. Intuitive description of a side-channel key recovery attack

In practice, side-channel attacks are usually divided in two phases. First an (optional) preparation phase provides the adversary with a training device and allows him to profile and characterize the leakages. Second, an exploitation phase is directly mounted against the target device and is aimed to succeed the key recovery. Importantly, actual adversaries do not always have the opportunity to carry out a preparation phase (in which case profiling is done on the fly). By contrast, it is an important phase for evaluators since it allows performing optimized attacks and therefore leads to a better analysis of the physical leakages. Before moving to the definitions of our metrics, we finally mention the "theory" and "practice" arrows leading to the information theoretic metric in Figure 1. These arrows underline the fact that one can always assume a theoretical model for the side-channel and perform a *simulated attack*. If the model is meaningful, so is the simulated attack. But such simulations always have to be followed by an *experimental attack* in order to confirm the relevance of the model. Experimental attacks exploit actual leakages obtained from a measurement setup.

3 Formal Definitions

In this section, we define the metrics that we suggest for the analysis of physically observable devices. We first detail two possible security metrics, corresponding to different computational strategies. Both metrics relate to the notion of side-channel key recovery. Then, we propose an information theoretic metric driven by two requirements: (1) being independent of the adversary and (2) having the same meaning for any implementation or countermeasure. As a matter of fact and following the standard approach in information theory, Shannon's conditional entropy is a good candidate for such a metric. Typically, the use of an average criteria to compare implementations is justified by the need of adversary independence. By contrast, the interactions of an adversary with a leaking system (*e.g.* adaptive strategies) are quantified with the security metrics in our model. We note that these candidate metrics will be justified by theoretical facts in Section 5 and practical applications in Section 6. However, it is an interesting open problem to determine if other metrics are necessary to evaluate side-channel attacks (*e.g.* min entropy is briefly discussed in Section 6).

3.1 Actual Security Metrics

Success Rate of the Adversary. Let $\mathsf{E}_K = \{\mathsf{E}_k(.)\}_{k\in\mathcal{K}}$ be a family of cryptographic abstract computers indexed by a variable key K. Let $(\mathsf{E}_K, \mathsf{L})$ be the physical computers corresponding to the association of E_K with a leakage function L. As most cryptanalytic techniques, side-channel attacks are usually based on a divide-and-conquer strategy in which different (computationally tractable) parts of a secret key are recovered separately. In general, the attack defines a function $\gamma : \mathcal{K} \rightarrow \mathcal{S}$ which maps each key k onto an equivalent key class[1] $s = \gamma(k)$, such

[1] We focus on recovering key bytes for simplicity and because they are usual targets in side-channel attacks. But any other intermediate value in an implementation could be recovered, *i.e.* in general we can choose $s = \gamma(k, x)$ with x the input of $\mathsf{E}_k(.)$.

that $|\mathcal{S}| \ll |\mathcal{K}|$. We define a side-channel key recovery adversary as an algorithm $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ with time complexity τ, memory complexity m and q queries to the target physical computer. Its goal is to guess a key class $s = \gamma(k)$ with non negligible probability, by exploiting its collected (black box and physical) information. For this purpose, we assume that the output of the adversary $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ is a guess vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{S}|}]$ with the different key candidates sorted according to the attack result: the most likely candidate being g_1. A practice-oriented description of $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ with a detailed specification of its features is given in [30], Appendix A. Finally, we define a side-channel key recovery of order o with the experiment:

Experiment $\mathbf{Exp}^{\text{sc-kr-}o}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}$
$[\mathbf{g} \leftarrow \mathsf{A}_{\mathsf{E}_k,\mathsf{L}};\ s = \gamma(k);\ k \xleftarrow{R} \mathcal{K};]$
if $s \in [g_1, \ldots, g_o]$ **then** return 1;
else return 0;

The o^{th}-order success rate of $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ against a key class variable S is defined as:

$$\mathbf{Succ}^{\text{sc-kr-}o,S}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}(\tau, m, q) = \Pr\ [\mathbf{Exp}^{\text{sc-kr-}o}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}} = 1] \tag{1}$$

Intuitively, a success rate of order 1 (*resp.* 2, ...) relates to the probability that the correct key is sorted first (*resp.* among the two first ones, ...) by the adversary. When not specified, a first order success rate is assumed.

Computational Restrictions. Similarly to black box security, computational restrictions have to be imposed to side-channel adversaries in order to capture the reality of physically observable cryptographic devices. This is the reason for the parameters τ, m, q. Namely, the attack time complexity τ and memory complexity m (mainly dependent on the number of key classes $|\mathcal{S}|$) are limited by present computer technologies. The number of measurement queries q is limited by the adversary's ability to monitor the device. In practice, these quantities are generally separated for the preparation and exploitation phases (see Section 5). But additionally to the computational cost of the side-channel attack itself, another important parameter is the remaining workload after the attack. For example, considering a success rate of order o implies that the adversary still has a maximum of o key candidates to test after the attack. If this has to be repeated for different parts of the key, it may become a non negligible task. As a matter of fact, the previously defined success rate measures an adversary with a fixed maximum workload after the side-channel attack. A more flexible metric that is also convenient in our context is the guessing entropy. It measures the average number of key candidates to test after the side-channel attack. The guessing entropy was originally defined in [23] and has been proposed to quantify the effectiveness of adaptive side-channel attacks in [20]. It can be related to the notion of gain that has been used in the context of multiple linear cryptanalysis to measure how much the complexity of an exhaustive key search is reduced thanks to an attack [5]. We use it as an alternative to the success rate.

Guessing Entropy. We first define a side-channel key guessing experiment:

$$\begin{array}{l}\text{Experiment } \mathbf{Exp}^{\text{sc-kg}}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}} \\ [\mathbf{g} \leftarrow \mathsf{A}_{\mathsf{E}_k,\mathsf{L}};\ s = \gamma(k);\ k \xleftarrow{R} \mathcal{K};] \\ \text{return } i \text{ such that } g_i = s;\end{array}$$

The guessing entropy of $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ against a key class variable S is then defined as:

$$\mathbf{GE}^{\text{sc-kr-}S}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}(\tau, m, q) = \mathbf{E}\big(\mathbf{Exp}^{\text{sc-kg}}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}\big) \tag{2}$$

3.2 Information Theoretic Metric

Let S be the previously used target key class discrete variable of a side-channel attack and s be a realization of this variable. Let $\mathbf{X}_q = [X_1, X_2, \ldots, X_q]$ be a vector of variables containing a sequence of inputs to the target physical computer and $\mathbf{x}_q = [x_1, x_2, \ldots, x_q]$ be a realization of this vector. Let $\mathbf{L}_q$ be a random vector denoting the side-channel observations generated with q queries to the target physical computer and $\mathbf{l}_q = [l_1, l_2, \ldots, l_q]$ be a realization of this random vector, *i.e.* one actual output of the leakage function L corresponding to the input vector $\mathbf{x}_q$. Let finally $\Pr[s|\mathbf{l}_q]$ be the conditional probability of a key class s given a leakage $\mathbf{l}_q$. We define the conditional entropy matrix as:

$$\mathbf{H}^q_{s,s^*} = -\sum_{\mathbf{l}_q} \Pr[\mathbf{l}_q|s] \cdot \log_2 \Pr[s^*|\mathbf{l}_q], \tag{3}$$

where s and s^* respectively denote the correct key class and a candidate out of the $|\mathcal{S}|$ possible ones. From 3, we derive Shannon's conditional entropy:

$$\mathrm{H}[S|\mathbf{L}_q] = -\sum_{s} \Pr[s] \sum_{\mathbf{l}_q} \Pr[\mathbf{l}_q|s] \cdot \log_2 \Pr[s|\mathbf{l}_q] = \mathop{\mathbf{E}}_{s}\left(\mathbf{H}^q_{s,s}\right) \tag{4}$$

It directly yields the mutual information: $\mathrm{I}(S;\mathbf{L}_q) = \mathrm{H}[S] - \mathrm{H}[S|\mathbf{L}_q]$. Note that the inputs and outputs of an abstract computer are generally given to the side-channel adversary (but hidden in the formulas for clarity reasons). Therefore, it is implicitly a computational type of entropy that is proposed to evaluate the physical leakages. This is because divide-and-conquer strategies target a key class assuming that the rest of the key is unknown. But from a purely information theoretic point of view, the knowledge of a plaintext-ciphertext pair can determine a key completely (*e.g.* for block ciphers). Hence and as detailed in the next section, the amount of information extracted by a side-channel adversary depends on its computational complexity. Note also that leakage functions can be discrete or (most frequently) continuous. In the latter case, it is formally a conditional differential entropy that is computed. Note finally that in simulated attacks where an analytical model for a continuous leakage probability distribution is assumed, the previous sums over the leakages can be turned into integrals.

4 Practical Limitations

One important goal of the present framework is to allow a sound evaluation of any given implementation, if possible independently of an adversary's algorithmic details. For this purpose, the strategy we follow is to consider an information theoretic metric that directly depends on the leakages probability distribution $\Pr[\mathbf{L}_q|S]$. Unfortunately, there are two practical caveats in this strategy.

First, the conditional probability distribution $\Pr[\mathbf{L}_q|S]$ is generally unknown. It can only be approximated through physical observations. This is the reason for the leakage function abstraction in the model of Micali and Reyzin that we follow in this work. It informally states that the only way an adversary knows the physical observables is through measurements. Therefore, practical attacks and evaluations have to exploit an approximated distribution $\hat{\Pr}[\mathbf{L}_q|S]$ rather than the actual one $\Pr[\mathbf{L}_q|S]$. Second, actual leakages may have very large dimensions since they are typically the output of a high sampling rate acquisition device like an oscilloscope. As a consequence, the approximation of the probability distribution for all the leakage samples is computationally intensive. Practical attacks usually approximate the probability distribution of a reduced set of samples, namely $\hat{\Pr}[\tilde{\mathbf{L}}_q|S]$. We denote side-channel attacks that exploit the approximated probability distribution of a reduced set of leakage samples as generic template attacks. A straightforward consequence of the previous practical limitations is that for any actual device, the mutual information $\mathrm{I}(S;\mathbf{L}_q)$ can only be approximated through statistical sampling, by using generic template attacks.

We note there are different concerns in the application of template attacks such as: "how to limit the number of leakage samples for which the distribution will be estimated?" or "how to limit the number of templates to build?". The data dimensionality reduction techniques used in [4,32] and the stochastic models in [16,27] can be used to answer these questions in a systematic manner. But there is no general theory allowing one to decide what is the best attack for a given device. Hence, in the following we will essentially assume that one uses the "best available tool" to approximate the leakage distribution. Quite naturally, the better generic template attacks perform in practice, the better our framework allows analyzing physical information leakages.

5 Relations between the Evaluation Metrics

In this section, we provide theoretical arguments that justify and connect the previous information theoretic and security metrics. These connections allow us to put forward interesting features and theoretical limitations of our model. In particular, we will consider two important questions.

First, as mentioned in Section 4, generic template attacks require to estimate the leakage probability distribution. Such a leakage model is generally built during a preparation phase and then used to perform a key recovery during an exploitation phase (as pictured in Figure 1). And as mentioned in Section 3.1, these phases have to be performed within certain computational limits. Hence,

to the previously defined complexity values τ, m, q of the online phase, one has to add the complexities of the preparation phase, denoted as τ_p, m_p, q_p. The first question we tackle is: given some bounds on (τ_p, m_p, q_p), can an adversary build a good estimation of the leakage distribution? We show in Section 5.1 that the conditional entropy matrix of Equation (3) is a good tool to answer this question. We also show how it relates to the asymptotic success rate of a Bayesian adversary. Then, assuming that one can build a good approximation for the leakage distribution, we investigate the extent to which the resulting estimation of the mutual information allows comparing different implementations. Otherwise said, we analyze the dependencies between our information theoretic and security metrics. We show that there exist practically meaningful contexts of Gaussian side-channels for which strong dependencies can be put forward. But we also emphasize that no general statements can be made for arbitrary distributions. Section 5.2 essentially states that the mutual information is a good metric to compare different implementations, but it always has to be completed with a security analysis (*i.e.* success rate and/or guessing entropy).

5.1 Asymptotic Meaning of the Conditional Entropy: "Can I Approximate the Leakage Probability Distribution?"

We start with three definitions.

Definition 1. The asymptotic success rate of a side-channel adversary $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ against a key class variable S is its success rate when the number of measurement queries q tends to the infinity. It is denoted as: $\mathbf{Succ}^{\text{sc-kr-}o,S}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}(q \to \infty)$.

Definition 2. Given a leakage probability distribution $\Pr[\mathbf{L}_q|S]$ and a number of side-channel queries stored in a leakage vector $\mathbf{l}_q$, a Bayesian side-channel adversary is an adversary that selects the key as $argmax_{s^*} \Pr[s^*|\mathbf{l}_q]$.

Definition 3. An approximated leakage distribution $\hat{\Pr}[\tilde{\mathbf{L}}_q|S]$ is sound if the first-order asymptotic success rate of a Bayesian side-channel adversary exploiting this leakage distribution against the key class variable S equals one.

In this section, we assume that one has built an approximated leakage distribution $\hat{\Pr}[\tilde{\mathbf{L}}_q|S]$ with some (bounded) measurement queries q_p, memory m_p and time τ_p. We want to evaluate if this approximation is good. For theoretical purposes, we consider an adversary/evaluator who can perform unbounded queries to the target device during the exploitation phase. We use these queries to evaluate the entropy matrix $\widehat{\mathbf{H}}^q_{s,s^*}$ defined in Section 3.2. It directly leads to the following relation with the asymptotic success rate of a Bayesian adversary.

Theorem 1. *Assuming independent leakages for the different queries in a side-channel attack, an approximated leakage probability distribution* $\hat{\Pr}[\tilde{\boldsymbol{L}}_q|S]$ *is sound if and only if the conditional entropy matrix evaluated in an unbounded exploitation phase is such that* $argmin_{s^*} \widehat{\boldsymbol{H}}^q_{s,s^*} = s, \forall s \in \mathcal{S}$.

The proof of Theorem 1 is given in [30]. There are several important remarks:

1. Theorem 1 only makes sense for bounded preparation phases. For unbounded preparations, an adversary would eventually access the exact distribution $\Pr[\mathbf{L}_q|S]$. In this context, the soundness does only depend on the cardinality of the different sets $\{s^* | \Pr[\mathbf{L}_q|s^*] = \Pr[\mathbf{L}_q|s]\}$, $\forall s \in \mathcal{S}$.
2. The condition of independence for consecutive leakages is not expected to be fully verified in practice. For example, there could exist history effects in the side-channel observations. However, it is expected to hold to a sufficient degree for our proof to remain meaningful in most applications.
3. In practice, the exploitation phase in a side-channel attack is bounded as the preparation. Therefore, Theorem 1 will be relevant as long as the number of leakages used to test the approximated leakage distribution and estimate the conditional entropy matrix is sufficient.
4. Finally, the condition on the entropy matrix $\widehat{\mathbf{H}}^q_{s,s^*}$ is stated for the number of queries q for which the leakage distribution $\Pr[\mathbf{L}_q|S]$ was approximated during the preparation phase. In general, finding a sound approximation for q implies that it should also be feasible to find sound approximations for any $q' > q$. But in practice, computational limitations can make it easier to build a sound approximation for small q values than for larger ones.

5.2 Comparative Meaning of the Conditional Entropy: "Does More Entropy Imply More Security?"

Let us write an exemplary conditional entropy matrix and its estimation as:

$$\mathbf{H}^q_{s,s^*} = \begin{pmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,|\mathcal{S}|} \\ h_{2,1} & h_{2,2} & \dots & h_{2,|\mathcal{S}|} \\ \dots & \dots & \dots & \dots \\ h_{|\mathcal{S}|,1} & h_{|\mathcal{S}|,2} & \dots & h_{|\mathcal{S}|,|\mathcal{S}|} \end{pmatrix} \quad \widehat{\mathbf{H}}^q_{s,s^*} = \begin{pmatrix} \hat{h}_{1,1} & \hat{h}_{1,2} & \dots & \hat{h}_{1,|\mathcal{S}|} \\ \hat{h}_{2,2} & \hat{h}_{2,2} & \dots & \hat{h}_{2,|\mathcal{S}|} \\ \dots & \dots & \dots & \dots \\ \hat{h}_{|\mathcal{S}|,1} & \hat{h}_{|\mathcal{S}|,2} & \dots & \hat{h}_{|\mathcal{S}|,|\mathcal{S}|} \end{pmatrix}$$

Theorem 1 states that if the diagonal values of a (properly approximated) matrix are minimum for all key classes $s \in \mathcal{S}$, then these key classes can be asymptotically recovered by a Bayesian adversary. As a matter of fact, it gives rise to a binary conclusion about the approximated leakage probability distribution. Namely, Theorem 1 answers the question: "*Can one approximate the leakage probability distribution under some computational bounds τ_p,m_p,q_p?*".

Let us now assume that the answer is positive and denote each element $h_{s,s}$ as the residual entropy of a key class s. In this subsection, we are interested in the values of these entropy matrix elements. In particular, we aim to highlight the relation between these values and the effectiveness of a side-channel attack, measured with the success rate. Otherwise said, we are interested in the question: "*Does less entropy systematically implies a faster convergence towards a 100% success rate?*". Contrary to the previous section, this question makes sense both for the ideal conditional entropy matrix that would correspond to an exact leakage distribution and for its approximation. Since general conclusions for arbitrary leakage distributions are not possible to obtain, our strategy is to

first consider simple Gaussian distributions and to extrapolate the resulting conclusions towards more complex cases. We start with three definitions.

Definition 4. An $|\mathcal{S}|$-target side-channel attack is an attack where an adversary tries to identify one key class s out of $|\mathcal{S}|$ possible candidates.

Definition 5. An univariate (*resp.* multivariate) leakage distribution is a probability distribution predicting the behavior of one (*resp.* several) leakage samples.

Definition 6. A Gaussian leakage distribution is the probability distribution of a leakage function that can be written as the sum of a deterministic part and a normally distributed random part, with mean zero and standard deviation σ.

Finally, since we now consider the residual entropies of the different key classes separately, we need a more specific definition of the success rate against a key class s (*i.e.* a realization of the variable S), denoted as $\mathbf{Succ}^{\text{sc-kr-}o,s}_{\mathsf{A}_{\mathsf{E}_k},\mathsf{L}}(\tau, m, q)$. It corresponds to the definition of Section 3.1 with a fixed key class.

Examples. Figure 2 illustrates several Gaussian leakage distributions. The upper left picture represents the univariate leakage distributions of a 2-target side-channel attack, each Gaussian curve corresponding to one key class s. The upper right picture represents the bivariate leakage distributions of a 2-target side-channel attack. Finally, the lower left and right pictures represent the univariate and bivariate leakage distributions of an 8-target side-channel attack. Note that in general, the internal state of an implementation does not only depend on the keys but also on other inputs, *e.g.* the plaintexts in block ciphers. Hence, the different dimensions in a multivariate distribution can represent both the different samples of a single leakage trace (generated with a single plaintext) or different traces (*e.g.* each dimension could correspond to a different plaintext). Eventually, it is an adversary's choice to select the internal states for which templates will be built. Therefore, we do not claim that these distributions always connect to practical attacks. But as will be seen in the following, even these simple theoretical contexts hardly allow simple connections between information and security.

We now discuss formally the connections between the success rate against a key class s and its residual entropy for idealized distributions and attacks.

Definition 7. An ideal side-channel attack is a Bayesian attack in which the leakages are exactly predicted by the adversary's approximated probability density function $\hat{\Pr}[\tilde{\mathbf{L}}_q|S]$ (*e.g.* thanks to an unbounded preparation phase).

Lemma 1. *In an ideal 2-target side-channel attack exploiting a univariate Gaussian leakage distribution, the residual entropy of a key class s is a monotonously decreasing function of the single query (hence multi-queries) success rate against s.*

Lemma 2. *In an ideal 2-target side-channel attack exploiting a multivariate Gaussian leakage distribution, with independent leakage samples having the same noise standard deviation, the residual entropy of a key class s is a monotonously decreasing function of the single query (hence multi-queries) success rate against s.*

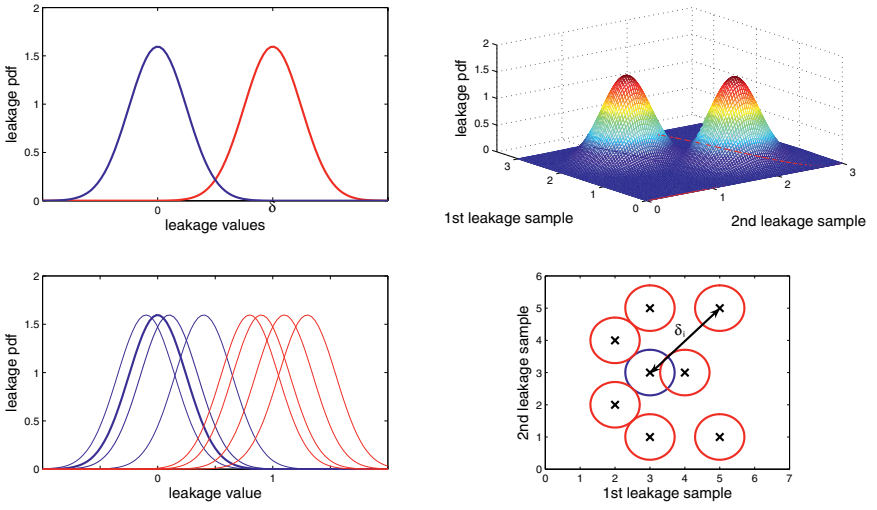


Fig. 2. Illustrative leakage probability distributions $\Pr[\mathbf{L}_q|S]$

These lemmas essentially state that (under certain conditions) the entropy and success rate in a 2-target side-channel attack only depend on the distance between the target leakages mean values normalized by their variance. It implies the direct intuition that more entropy means less success rate. Unfortunately, when moving to the $|\mathcal{S}|$-target case with $|\mathcal{S}| > 2$, such a perfect dependency does not exist anymore. One can observe in the lower right part of Figure 2 that the entropy and success rate not only depend on the normalized distances δ_i/σ but also on how the keys are distributed within the leakage space. Therefore, we now define a more specific context in which formal statements can be proven.

Definition 8. A perfect Gaussian leakage distribution $\Pr[\mathbf{L}_q|s]$ for a key class s is a Gaussian leakage distribution with independent leakage samples having the same noise standard deviation such that the Euclidean distance between each key class candidate mean value and the correct key class candidate mean value is equal and the residual entropy of the key class s is maximum.

Theorem 2. *In an ideal side-channel attack exploiting a perfect Gaussian leakage distribution, the residual entropy of a key class s is a monotonously decreasing function of the single query (hence multi-queries) success rate against s.*

The proofs of Lemmas 1, 2 and Theorem 2 are given in [30]. They constitute our main positive results for the use of the conditional entropy as a comparison metric for different implementations. Unfortunately, in the most general context of non perfect leakage distributions, those general statements do not hold. Facts 1 and 2 in [30] even demonstrate that there exist no generally true dependencies between the conditional entropy and the success rate in a general setting.

5.3 Intuition of the Metrics

In this section, we recall and detail a number of important intuitions that can be extracted from the previous theory. We also discuss how they can be exploited in practical applications and highlight their limitations.

Intuitions Related to Theorem 1

1.1 *Theorem 1 tells if it is possible to approximate a given leakage function in a bounded preparation phase.* As mentioned in Section 4, such an approximation highly depends on the actual tools that are used for this purpose. In general, the better the tools, the better the evaluation. Hence, Theorem 1 allows checking if these tools are powerful enough. If they are not...

1.2 *Theorem 1 indicates some resistance of the target implementation against side-channel attacks.* If one cannot build a sound approximation of the leakage probability distribution, even with intensive efforts, then the 1^{st}-order asymptotic success rate of the Bayesian side-channel adversary does not reach one. But this does not imply security against side-channel attacks (*e.g.* think about a device where only one key could not be recovered). In this context, it is important to evaluate the actual security metrics for different adversaries in order to check if high success rates can still be reached.

Intuitions Related to Theorem 2

2.1 *Theorem 2 only applies to sound leakage distributions.* Intuitively, it means that comparing the conditional entropy provided by different leakage functions only makes sense if the corresponding approximated leakage probability distribution lead to asymptotically successful attacks.

2.2 *Theorem 2 confirms that mutual information is a relevant tool to compare different implementations.* It shows meaningful contexts of Gaussian channels for which less residual entropy for a key class implies a more efficient attack. It strengthens the intuitive requirements of Section 3, namely an adversary independent metric with the same meaning for any implementation.

2.3 *The conditional entropy is not a stand-alone metric to compare implementations and always has to be combined with a security analysis.* This relates both to theoretical limitations (since there exists no general relation between information and security) and practical constraints. For a given amount of information leaked by an implementation, different side-channel distinguishers could be considered. Therefore, security metrics are useful to evaluate the number of queries for these different attacks to succeed.

Note that the mutual information, success rates and guessing entropy are average evaluation criteria. However in practice, the information leakages and security of an implementation could be different for different keys. Therefore, it is important to also consider these notions for the different keys separately (*e.g.* to evaluate

the conditional entropy matrix rather than the mutual information). This last remark motivates the following practice-oriented definition.

Definition 9. We say that a side-channel attack against a key class variable S is a weak template attack if all the key classes s have the same residual entropy $h_{s,s}$ and each line of the entropy matrix $\mathbf{H}^q_{s,s^*}$ is a permutation of another line of the matrix. We say that a side-channel attack is a strong template attack if at least one of the previous conditions does not hold.

6 Applications of the Model

In order to confirm that (although limited by theoretical concerns) the intuition of Theorem 2 applies to practice, this section provides examples of side-channel attacks that can be reproduced by the reader. Applications to more complex and practically meaningful contexts can be found in other publications [21,30,31,32,33].

For this purpose, we consider a known plaintext attack against a reduced block cipher that we formalize as follows. Let S be a 4-bit substitution box, *e.g.* from the AES candidate Serpent. We target the computation of $y = \mathsf{S}(x \oplus k)$, where x is a random plaintext and k a secret key. A Bayesian adversary is provided with observations $(x, \mathsf{L}'(y) + r)$, where r is a gaussian noise with mean 0 and standard deviation σ. For any y value, the deterministic part of the leakage $\mathsf{L}'(y)$ is given by a vector $\mathbf{Z}$. The adversary's goal is to recover the key k. Our simulations exploit different leakage functions and assume an unbounded preparation phase (*i.e.* the adversary has knowledge of the exact leakage distribution). We start the frequently observed Hamming weight leakages and $\mathbf{Z}_1 = [0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4]$. We also evaluate two other leakage functions represented by the vectors: $\mathbf{Z}_2 = [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3]$ and $\mathbf{Z}_3 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 3, 3, 4, 4]$. The conditional entropies and single-query success rates with $\sigma = 0$ can be straightforwardly computed as:

$$\mathrm{H}[K|\mathbf{L}_1^{\mathbf{Z}_1}] \simeq 1.97 \qquad \mathrm{H}[K|\mathbf{L}_1^{\mathbf{Z}_2}] = 2 \qquad \mathrm{H}[K|\mathbf{L}_1^{\mathbf{Z}_3}] \simeq 2.16$$
$$\mathbf{Succ}^{\text{sc-kr}}_{\mathbf{L}_1^{\mathbf{Z}_1}}(q=1) = \tfrac{5}{16} \qquad \mathbf{Succ}^{\text{sc-kr}}_{\mathbf{L}_1^{\mathbf{Z}_2}}(q=1) = \tfrac{1}{4} \qquad \mathbf{Succ}^{\text{sc-kr}}_{\mathbf{L}_1^{\mathbf{Z}_3}}(q=1) = \tfrac{5}{16}$$

At first sight, it seems that these leakage functions exactly contradict Theorem 2. For example, when moving from $\mathbf{Z}_2$ to $\mathbf{Z}_3$, we see that both the conditional entropy and the success rate are increased. However, the goal of side-channel attacks is generally to reach high success rates that are not obtained with a single query. Hence, it is also interesting to investigate the success rate for more queries. In the left part of Figure 3, these success rates for increasing q values are plotted. It clearly illustrates that while $\mathbf{Z}_2$ leads to a lower success rate than $\mathbf{Z}_3$ for $q = 1$, the opposite conclusion holds when increasing q. That is, the intuition given by Theorem 2 only reveals itself for $q > 2$. Importantly, these conclusions can vary when noise is inserted in the leakages, *e.g.* assuming $\sigma = 1$, we have:

$$\mathrm{H}[K|\mathbf{L}_1^{\mathbf{Z}_1}] \simeq 3.50 \qquad \mathrm{H}[K|\mathbf{L}_1^{\mathbf{Z}_2}] \simeq 3.42 \qquad \mathrm{H}[K|\mathbf{L}_1^{\mathbf{Z}_3}] \simeq 3.22$$

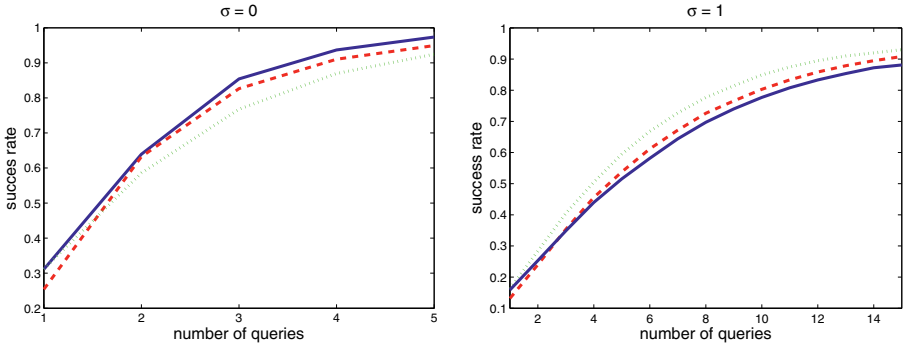


Fig. 3. 1st-Order success rates in function of the number of queries for the leakages functions corresponding to $\mathbf{Z}_1$ (solid line), $\mathbf{Z}_2$ (dashed line) and $\mathbf{Z}_3$ (dotted line)

The right part of Figure 3 plots the success rates of these noisy leakage functions. It again highlights a context in which Theorem 2 is eventually respected. In general, these examples underline another important feature of our metrics. Namely, the more challenging the side-channel attack (*i.e.* the more queries needed to reach high success rates), the more significant the conditional entropy is. Otherwise said: the mutual information better reveals its intuition asymptotically. And in such contexts, the single-query success rate can be misleading.

Note that the examples in this section are more directly reflective of actual side-channel attacks in which different plaintexts can generally be used to identify a key class than the ideal contexts investigated in Section 5.2.

A Short Note on Minimum Entropy. With respect to the relevance of other metrics in the model, we finally mention that min entropy is equivalent to a single-query success rate. Since side-channel attacks are essentially multiple-query attacks, we believe that Shannon's conditional entropy better captures the information leakages in most practical applications. For example, Figure 3 is typical of contexts where min entropy is misleading, *i.e.* where the success rate for $q = 1$ is not very significant while the conditional entropy nicely quantifies the evolution of this success rate for any larger q. But as already said, the information theoretic analysis always has to be completed with a security analysis. Hence, even in contexts where min entropy is the right metric, our model would detect it.

7 Evaluation Methodology

Following the previous sections, an evaluation methodology for side-channel attacks intends to analyze both the quality of an implementation and the strength of an adversary, involving the five steps illustrated in Figure 4. It again indicates that the information theoretic metric can be used to measure an implementation while the actual security metrics are rather useful to evaluate adversaries. Additionally to these metrics, it is often interesting to define a Signal-to-Noise Ratio (SNR) in order to determine the amount of noise in the physical observations.

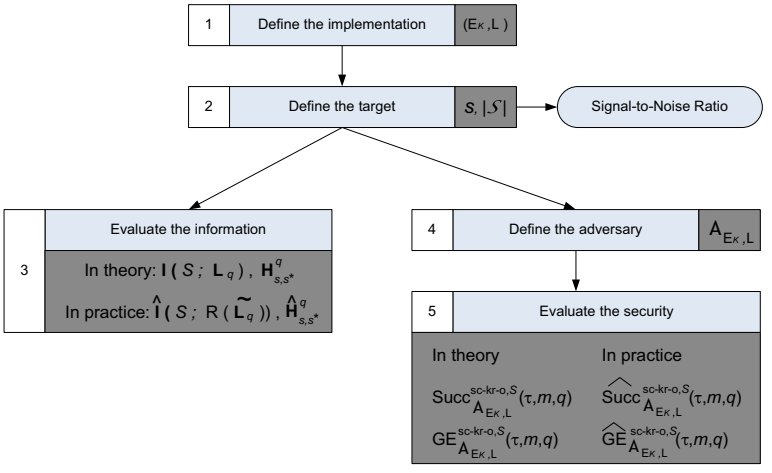


Fig. 4. Evaluation methodology for side-channel attacks

Since noise insertion is a generic countermeasure to improve resistance against side-channel attacks, it can be used to plot the information theoretic and security metrics with its respect. We note finally that the definition of an implementation requires to evaluate the cost of the equipment used to monitor the leakages. Since quantifying such costs is typically the tasks assigned the standardization bodies, we refer to the common criteria [9] and FIPS 140-2 documents [13] (or alternatively to the IBM taxonomy [1]) for these purposes. In general, the benefit of the present model is not to solve these practical issues but to state the side-channel problem in a sound framework for its analysis. Namely, it is expected that the proposed security and information theoretic metrics can be used for the fair analysis, evaluation and comparison of any physical implementation or countermeasure against any type of side-channel attack.

8 Conclusions and Open Problems

A framework for the analysis of cryptographic implementations is introduced in order to unify the theory and practice of side-channel attacks. It is aimed to bridge the formal understanding of physically observable cryptography to the exploitation of actual leakages in experimental key recoveries. The framework is centered around a theoretical model in which the effect of practically relevant leakage functions is evaluated with a combination of security and information theoretic metrics. It allows discussing the underlying tradeoffs in physically observable cryptography in a fair manner. As an interface between an engineering problem (how much is leaked?) and a cryptographic problem (how to exploit it?), our framework helps putting forward properly quantified weaknesses in physically observable devices. The fair evaluations that it provides can then be used in two directions. Either the physical weaknesses can be sent to hardware designers

in order to reduce physical leakages. Or they can be transmitted to cryptographic designers in order to conceive schemes that can cope with physical leakages.

Open questions derive from this model in different directions. A first one relates to the best exploitation of large side-channel traces, *i.e.* to the construction of (ideally) optimal distinguishers. This requires investigating the best heuristics to deal with high dimensional leakage data (our model assumes adversaries exploiting such specialized algorithms). A second one relates to the investigation of stronger security notions than side-channel key recovery. That is, the different security notions considered in the black box model (*e.g.* undistinguishability from an idealized primitive) should be considered in the physical world, as initiated in [24] (but again in a more specialized way). A third possible direction relates to the construction of implementations with provable (or arguable) security against side-channel attacks, *e.g.* as proposed in [11,26,25]. Finally, this work could be extended to other physical threats (*e.g.* fault attacks) and combined with other approaches for modeling physical attacks such as [15,17,18].

References

1. Abraham, D.G., Dolan, G.M., Double, G.P., Stevens, J.V.: Transaction Security System. IBM Systems Journal 30(2), 206–229 (1991)
2. Agrawal, D., Archambeault, B., Rao, J., Rohatgi, P.: The EM side-channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
3. Backes, M., Köpf, B.: Formally Bounding the Side-Channel Leakage in Unknown-Message Attacks, IACR ePrint archive (2008), `http://eprint.iacr.org/2008/162`
4. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
5. Biryukov, A., De Cannière, C., Quisquater, M.: On multiple linear approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
6. Cachin, C.: Entropy Measures and Unconditional Security in Cryptography, PhD Thesis, ETH Dissertation, num 12187, Zurich, Switzerland (1997)
7. Chari, S., Rao, J., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
8. Cryptographic Hardware and Embedded Systems, `http://www.chesworkshop.org`
9. Application of Attack Potential to Smart Cards, Common Criteria Supporting Document, Version 1.1 (July 2002), `http://www.commoncriteriaportal.org`
10. Cover, T.M., Thomas, J.A.: Information Theory. Wiley and Sons, New York (1991)
11. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: The proceedings of FOCS 2008, Philadelphia, USA, pp. 293–302 (October 2008)
12. ECRYPT Network of Excellence in Cryptology, The Side-Channel Cryptanalysis Lounge, `http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html`
13. FIPS 140-2, Security Requirements for Cryptographic Modules, Federal Information Processing Standard, NIST, U.S. Dept. of Commerce (December 3, 2002)
14. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)

15. Gennaro, R., Lysyanskaya, A., Malkin, T.G., Micali, S., Rabin, T.: Algorithmic Tamper-Proof Security: Theoretical Foundations for Security Against Tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
16. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic methods. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006)
17. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
18. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
19. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
20. Köpf, B., Basin, D.: an Information Theoretic Model for Adaptive Side-Channel Attacks. In: The proceedings of ACMCCS 2007, Alexandria, VA, USA (October 2007)
21. Macé, F., Standaert, F.-X., Quisquater, J.-J.: Information theoretic evaluation of side-channel resistant logic styles. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 427–442. Springer, Heidelberg (2007)
22. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)
23. Massey, J.L.: Guessing and Entropy. In: The proceedings of the IEEE International Symposium on Information Theory, Trondheim, Norway, p. 204 (June 1994)
24. Micali, S., Reyzin, L.: Physically observable cryptography. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
25. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T.G., Yung, M.: A Block Cipher based PRNG Secure Against Side-Channel Key Recovery. In: ASIACCS 2008, Tokyo, Japan, pp. 56–65 (March 2008)
26. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: The proceedings of Eurocrypt 2009, Cologne, Germany. LNCS (April 2009) (to appear)
27. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
28. Shannon, C.E.: A Mathematical Theory of Communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
29. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal 28, 656–715 (1949)
30. Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks (extended version), Cryptology ePrint Archive, Report 2006/139
31. Standaert, F.-X., Peeters, E., Archambeau, C., Quisquater, J.-J.: Towards security limits in side-channel attacks. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 30–45. Springer, Heidelberg (2006)
32. Standaert, F.-X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008)
33. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)

A Leakage-Resilient Mode of Operation

Krzysztof Pietrzak

CWI Amsterdam, The Netherlands

Abstract. A weak pseudorandom function (wPRF) is a cryptographic primitive similar to – but weaker than – a pseudorandom function: for wPRFs one only requires that the output is pseudorandom when queried on *random* inputs. We show that unlike "normal" PRFs, wPRFs are seed-incompressible, in the sense that the output of a wPRF is pseudorandom even if a bounded amount of information about the key is leaked.

As an application of this result we construct a simple mode of operation which – when instantiated with any wPRF – gives a *leakage-resilient* stream-cipher. The implementation of such a cipher is secure against *every* side-channel attack, as long as the amount of information leaked *per round* is bounded, but overall can be arbitrary large. The construction is simpler than the previous one (Dziembowski-Pietrzak FOCS'08) as it only uses a single primitive (a wPRF) in a straight forward manner.

1 Introduction

Traditionally, cryptographic algorithms are designed to withstand adversaries that can attack the cryptosystem in a black-box fashion. This means that all the adversary can do is to query the system at hand according to the security definition. In many settings this is not a realistic assumption, as real-world adversaries attack concrete *implementations* of cryptosystems, that possibly leak information which cannot be efficiently computed from black-box access alone. Attacks exploiting such leakage are called side-channel attacks. In the last two decades we saw many cryptanalytic attacks exploiting side-channels as running-time [31], electromagnetic radiation [39, 19], power consumption [33] and fault detection [4, 3]. A recent example [18] is the side-channel attack against KeeLoq (which refers to the "KeeLoq block-cipher" and some particular mode in which this cipher is used), which is widely used as e.g. anti-theft mechanisms for cars. Although the KeeLoq block-cipher seems not to be very secure to start with [9, 27], the devastating side-channel attack of [18] exploits a weakness in the mode in which the cipher is used, rather than a weakness in the cipher itself, and it would still be applicable even if the KeeLoq block-cipher was replaced with a strong block-cipher, say AES ([18] Talk of Christof Paar). It is thus an intriguing question whether there exist modes of operation which are provably secure against a wide class of side-channel attacks if instantiated with any block-cipher.

In this paper we answer this question affirmatively, by proposing a mode of operation (cf. Figure 1) which turns any weak PRF into a stream-cipher which is provably secure against *all* side-channel attacks, assuming only that the amount

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 462–482, 2009.

of leakage in each round is bounded, and that only memory which is actually accessed in some round leaks in this round. Such a "leakage-resilient" cipher was recently constructed in [17], the main advantage of our new construction is its simplicity, it can be instantiated with any weak PRF (e.g. with a block-cipher like AES), whereas the construction from [17] additionally required extractors.

The simplicity of the construction (as compared to [17]) comes at the price of more involved security proof. Besides the technical tools we already used in [17], we will need new results concerning the security of weak PRFs when neither the key nor the inputs are uniform. The technique we use to prove this results can also be applied in other settings, e.g. for encryption schemes, and thus could be of independent interest.

Why Leakage-Resilience. Leakage-resilience is an extremely strong security notion considering adversaries who can choose arbitrary leakage functions. To practitioners this may seem like an overkill, after all, why consider unrealistic side-channels which leak some very involved function of the state instead of using some ad-hoc countermeasures against "real" side-channels? A lesson cryptographers have learned in the last decades is that ad-hoc arguments usually result in insecure systems, and this very much applies to the young history of side-channel cryptanalysis. Implementing cryptographic algorithms in a straight forward way, will almost certainly make them very susceptible to side-channel attacks. Often – like in differential power analysis [33, 8] – such attacks extract a little bit of information in each evaluation, and then combine this information to get the secret key. Thus it is crucial that an implementation does not leak even small amounts of (useful) information. In contrast, "leakage-resilient" algorithms as considered in this work guarantee security even if in *each* invocation a *bounded* amount of *arbitrary* information is leaked.

We advocate the following approach to side-channel security: first cryptographers design a leakage-resilient algorithm C, with the guarantee that whenever you implement C such that in each invocation $\leq \lambda$ bits of information leak, the implementation is safe. This still leaves the task of implementing C such that the $\leq \lambda$ leakage bound is met.[1] The rationale here is that this task is clearly much more realistic than having to implement an algorithm in way where nothing leaks at all, as it would be necessary if the algorithm would come with no bound on the leakage that can be tolerated. (cf. Kocher [32] for a similar argument). It is only at this stage that one should consider using ad-hoc measures like masking or blinding, using special circuit designs, and so on. Cryptography seems to be of limited use at this stage, but a background on existing attacks and implementation details is helpful here, thus this task is something that should be left to security researchers and engineers.

Some Related Work. Most papers on side-channel security – like [31, 39, 19, 33, 4, 3] mentioned in the introduction – consider attacks and/or countermeasures

[1] Note that this is unavoidable, as when one cannot keep at least some uncertainty about the internal state, one cannot hope to get a secure implementation.

against a specific side-channel. From the papers considering general models for side-channel attacks, the work of Micali and Reyzin [35] on "physically observable cryptography" is particularly insightful and written in a language accessible to cryptographers. Their model is based on five "axioms", some of which are (more or less explicitly) used in our model.

Ishai et al. [29, 28] consider a model where the adversary can choose some wires in the circuit, and then learns the values carried by those wires during the computation. What makes their work exceptional is that they were the first to *prove* how to implement *any* algorithm secure against an interesting side-channel (i.e. probing attacks).[2] The field of exposure-resilient cryptography [11] considers the more restricting case where the adversary could learn some of the *input* bits.

Very recently [1] showed that some *particular* public-key encryption schemes are surprisingly robust against leakage: the scheme stays secure even if the min-entropy of the key is just a constant fraction of the min-entropy of a random key. We prove a similar result for *any* weak PRFs, but in order prove security even for keys with such low min-entropy, we need the weak PRF to be exponentially hard, whereas [1] can do so with some particular superpolynomial assumptions (learning with error and lattice assumptions).

Papers that consider constructions of *stream-ciphers* which withstand side-channel attacks (as in this work and [17]) include [32, 35, 36]. Kocher [32] considers a very simple construction where one simply iterates a hash function (SHA256 is suggested). This work is kept informal, with no proofs or even formal claims, but contains several interesting conceptual ideas. Micali and Reyzin [35] investigate *reductions* of side-channel resistant primitives, in particular they show that the Blum-Micali construction is secure, assuming the implementation of the underlying permutation already satisfies some strong form of side-channel security. The work which aims at a goal most similar to ours is Petit et al. [36]. They propose and analyze a block-cipher based construction, where security against sides-channels is achieved by making it hard to "combine" leakages from different rounds.[3] Their underlying model [41] is motivated by practical considerations, considering leakage-functions and attacks that have been successfully used to break systems. Compared to [36], we take a much more theoretical approach, our setting is more general and the underlying assumptions are weaker

[2] Formally, Ishai et al. do the following: let $t \geq 0$ be some constant and let $[X]$ denote a $(t+1)$ out of $(t+1)$ secret sharing of the value X. They construct a general compiler, which turns every circuit $G(.)$ into a circuit $G_t(.)$ (of size $t^2|G|$) such that $[G(X)] = G_t([X])$ for all inputs X, and moreover one does not learn any information on $G(X)$ even when given the value carried by any t wires in the circuit $G_t(.)$ while evaluating the input $[X]$. This transformation uses multiparty-computation, which is quite different from all other approaches we discuss here.

[3] By using a forward secure primitive, one can ensure that *past* keys cannot be combined with the current key, as they cannot even be computed. For *future* keys, this is more tricky, as the cipher itself must be able to efficiently derive that keys.

in several aspects.[4] The tools and techniques from [17] and this paper cannot be used to prove security of the constructions from [32, 35, 36] (or any other construction we are aware of), as those constructions are insecure against arbitrary leakage functions as considered in this work, even if the underlying primitives are ideal (e.g. Random oracles in [32] or ideal ciphers in [36]) and only one bit of information leaks per invocation of the underlying primitive. (but this does by no means mean that they are insecure against side-channels that arise in practice.)[5]

Some interesting recent results in settings which are similar or otherwise relevant to general models of side-channel security include [5], who show how to securely realize protocols when perfect deletion is not possible. Goldwasser et al. [22] construct "one-time programs" from simple hardware satisfying some weak form of side-channel security. Dodis and Wichs [12] solve the long standing open problem of two round authenticated key-agreement from non-uniform keys. (See the full version [37] for a more detailed discussion on those papers.)

1.1 Leakage-Resilient Cryptography

In this section we informally introduce and motivate the model of "leakage-resilient cryptography" from [17].

Consider some keyed cryptographic primitive CP. The most general side-channel attack against $\mathsf{CP}(S_0)$ – where S_0 denotes the secret initial state – is to allow an attacker to choose any leakage function f, which then is evaluated on the initial state S_0, and the adversary receives $f(S_0)$.[6] Clearly we cannot hope for any security at all here, as f could simply output the complete state $f(S_0) = S_0$. Thus, it is necessary to somehow restrict the range of the leakage function, we will consider functions with range $\{0,1\}^\lambda$, where $\lambda \ll |S_0|$ is some parameter. The idea to define the set of leakage functions by restricting the output length was inspired by the bounded-retrieval model [10, 14, 13, 6, 16], which in turn was inspired by the bounded-storage model [34, 15, 42].

[4] In particular 1. We prove security in the standard model, whereas [36] work in the ideal-cipher model 2. The security notion considered in [36] is key-recovery, whereas we use unpredictability (and, in a limited context, indistinguishability). 3. The leakage functions considered in [36] (namely Hamming weight or identity plus noise) are motivated by leakages observed in practice, whereas we bound only the amount, not the type of information leaked 4. Finally, and most importantly, our approach differs in how the observed leakage (cf. point 3.) can be exploited in order to break the security notion (cf. point 2.). [36] show that a so called template attack [7] cannot recover the key, whereas we prove security against every efficient adversary.

[5] A crucial requirement we need from the construction in order to prove leakage-resilience, is that the state can be split in (at least) two parts, and this parts evolve independently, in the sense that any interaction between them is public. Formally, one must be able to express the cipher as a process as in Lemma 5 in this paper.

[6] Here the leakage function is applied only to the state S_0, and not to any internal variables appearing in the computation. This can be done without loss of generality as all the internal variables are simply functions of the state S_0, and thus can be computed by f.

As the implementation of any cryptosystem will leak more information the longer it runs, we want to allow the attacker A to adaptively choose different leakage functions during the lifetime of the system. For this, we assume that CP runs in rounds (where a "round" is just some well defined part of the computation), and denote with S_i the state of CP after round i (to simplify the exposition we assume that the size of the state remains constant).

The attacker A we consider can adaptively choose a leakage function f_i before the ith round, and after round i receives $f_i(S_{i-1})$, i.e. the leakage function evaluated on the state at the beginning of round i. Unfortunately also here no security is possible beyond round t, where $t \cdot \lambda \geq |S_0|$, as A can simply define the f_i's such that $f_i(S_{i-1})$ will be some λ bits of S_t. (note that for $i \leq t$, f_i can compute the future state S_t from its input S_{i-1}.) After round t the attacker A has learned the entire state S_t, and no security is possible beyond this point.

Thus if we want security even after (much) more than $|S_0|$ bits have leaked, we need to further restrict the leakage functions. The restriction we use is one of the "axioms" from [35], and states that "only computation leaks information". This means that f_i does not get the entire state S_{i-1} as input, but only the part of the state that is actually accessed by CP in the ith round.

On Efficient Leakage Functions. As we consider a computational primitive, and the total leakage can be larger than the entire state, we can only allow *efficient* leakage functions.[7] This is not explicitly stated, but naturally comes up in the model, where the main result (Theorem 2) puts an upper bound on the size of a *circuit* computing the entire random experiment in which the cipher is attacked.

On (non)-Uniformity. Throughout, we always consider non-uniform adversaries.[8] In particular, our main reduction is non-uniform, which means we prove that if an adversary *exists* who breaks the stream-cipher, then an adversary (of related complexity) *exists* who breaks the underlying weak PRF. The only step in the proof where we need non-uniformity is a lemma from [2] which relates two types of pseudoentropy notions. As [2] also prove this lemma in a uniform setting (albeit which much worse parameters), it should be possible (though we didn't check the details) to make our reduction uniform, that is to show how to efficiently construct an adversary against the weak PRF from any adversary against the stream-cipher. (we refer to Goldreich's article [20] as to why such a reduction is desirable.)

[7] A computationally unbounded leakage function could simply compute and output the initial state from the output of the stream cipher. If one assumes that the total leakage is smaller than the key [1, 12], considering computationally unbounded leakage functions is meaningful.

[8] Recall that a uniform adversary can be modelled as a Turing-machine which as input gets a security parameter, whereas (more powerful) non-uniform adversaries will, for each security parameter, additionally get a different polynomial-length advice string. Equivalently, we can model non-uniform adversaries as a sequence of circuits (indexed by the security parameter), which is what we will do.

Relaxing Bounded Leakage. As described above, in each round we allow the adversary to choose any function f with range $\{0,1\}^\lambda$, and she then learns the leakage $f(S)$, where S is the state accessed in this round. Note that this also captures any efficient leakage function g, where there exists another (efficient) leakage function f with range $\{0,1\}^\lambda$ such that $S \to f(S) \to g(S)$ is a Markov chain and where one can efficiently sample $g(S)$ given $f(S)$ (as an adversary in our model can ask for $f(S)$, and then compute $g(S)$ himself). This e.g. covers the case where the leakage function outputs a noisy version of S.

We chose to work with bounded leakage as it is a very clean and intuitive model, but for the proof we actually only require that $f(S)$ does not contain more than λ bits of "useful" information on S. Formally, "useful" means that the HILL-pseudoentropy (a notion to be defined in Section 4) of S does not drop by much more than λ bits given $f(S)$. Unfortunately this most general notion is quite unintuitive to work with.[9]

Relaxing the "only computation leaks information" Axiom. The leakage function in round i gets as input only that part of the state which is accessed in that round. This translates to the requirement on the implementation that memory which is not accessed, must not leak at all. In our model and for our particular construction (and also [17]) allowing the adversary to choose a single leakage function f with λ bits output, and then giving her the leakage $f(S^+)$ (where with S^+ we denote the part of the state which is accessed and S^- denotes the remaining state) is equivalent to let her choose two function f' and f'' with $\lambda/2$ bits output respectively, and then output the leakage $f'(S^+)$ and $f''(S^-)$. Thus it is o.k. if the entire state leaks as long the leakage of S^+ and S^- is independent. In particular, we also get security against attacks which seem not to obey the "only computation leaks information" axiom, like the cold boot attack from [23] (see also [1]), who show how measure significant parts of a key that was stored on some memory, even after power is turned off.

1.2 Seed Incompressibility

As main new technical tools we prove bounds on the security of weak PRFs when the key (or the inputs) are not uniformly random as assumed in the security definition for weak PRFs.

Recall that the standard security notion for a pseudorandom function (PRF) $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^m$ requires that for a random key $k \in \{0,1\}^\kappa$ no efficient attacker can distinguish $\mathsf{F}(k,.)$ from a uniformly random function. Motivated by the question if random-oracles can (in some settings) be instantiated with efficient functions, Halevi et al. [24] investigate the question whether

[9] A special more intuitive case – which is still more general than bounded leakage – is to consider any (not necessarily) efficient leakage function g where there exists an efficient f with range $\{0,1\}^\lambda$, such that given $f(S)$ one can efficiently sample some "fake" leakage $\tilde{g}(S)$ where $[S, g(S)]$ is computationally indistinguishable from $[S, \tilde{g}(S)]$ (bounded leakage corresponds to $\tilde{g} = g$). Note that here the sampling algorithm only gets $f(S)$, whereas the distinguisher gets S.

"seed-incompressible" functions exist. They consider a setting where an adversary initially gets a "compressed key" $f(k)$ (where $f : \{0,1\}^\kappa \to \{0,1\}^\lambda$ and $\lambda < \kappa$). A simple observation is that by giving this extra input to an adversary, no function $\mathsf{F}(k,.)$ can possibly be a PRF, as $f(k)$ could e.g. encode the first λ bits of $\mathsf{F}(k,X)$ (for some fixed X), and thus $\mathsf{F}(k,.)$ becomes easily distinguishable from random.

In this paper we revisit the concept of seed incompressibility, but for *weak* pseudorandom functions (wPRF): F is a wPRF, if $\mathsf{F}(k,.)$ cannot be distinguished from random, if queried on *random* inputs. Thus an adversary gets to see $X_1,\ldots,X_q$ and $Z_1,\ldots,Z_q$, and then must guess whether $Z_i = \mathsf{F}(k,X_i)$ or $Z_i = \mathbf{R}(X_i)$ where $\mathbf{R}$ is a uniformly random function. Unlike for normal PRFs, for wPRFs it is not clear if and how a compressed seed $f(k)$ helps the distinguisher, e.g. now simply setting $f(k)$ to denote the λ first bits of $\mathsf{F}(k,X)$ for some fixed input X will not trivially break the security of $\mathsf{F}(k,.)$ as here the adversary cannot choose the inputs X for which she gets to see $\mathsf{F}(k,X)$.

Of course by leaking λ bits of the key, we must tolerate some security loss. In particular, if we use the trivial attack just described (leaking λ bits of $\mathsf{F}(k,X)$), the adversary can get "lucky", and one of the q queries $X_1,\ldots,X_q$ will hit the fixed input X. Because of that, the adversary has some extra advantage of roughly $q/2^n$ (compared to an adversary not getting $f(k)$). Further, if we assume that the best attack against F is brute-force search over the keyspace, then leaking λ bits of the key will degrade the security by a factor of 2^λ. As we prove in Lemma 2, it doesn't get much worse than that: if $\mathsf{F}(k,.)$ cannot be distinguished with advantage more than ϵ, then the advantage (against somewhat smaller adversaries) is still bounded by roughly $2^\lambda(\epsilon + q^2/2^{n+1})$ (here we set t from Lemma 2 to n, and assume that n is large enough so that the last term in (3) can be ignored.)

We actually do not consider the setting where the key k is random, and then $f(k), |f(k)| = \lambda$ is leaked, but the more general case where k is sampled from some distribution with min-entropy at least $|k| - \lambda$. (and we need this more general case later when proving the security of the leakage-resilient stream-cipher), as for any function f and uniformly random k, k has still (expected) min-entropy at least $|k| - \lambda$ given $f(k)$.

We then prove a similar result (Lemma 3) concerning the security of wPRFs assuming the inputs (as opposed to the key) are not uniformly random.

Proof Sketch. We show that any wPRF is secure even when the secret key is only sampled from some distribution with min-entropy $|k| - \lambda$ by a (uniform) reduction. Assume an adversary A can distinguish $\mathsf{F}(k,.)$ from a random function (when queried on random inputs $X_1,\ldots,X_q$) with advantage ϵ'. Using the Markov bound one can show that this implies that a key k sampled from the above distribution is "weak" with probability at least $\epsilon'/2$, where a key k is said to be weak, if the distinguishing advantage of A, conditioned on the key being k, is at least $\epsilon'/2$. If k is now sampled from the uniform distribution (and not a distribution with min-entropy $|k| - \lambda$), then k will be weak with probability at least $\epsilon'/2^{\lambda+1}$, i.e. we loose at most a factor 2^λ. The crucial point is that

when observing the output of a function $g(.)$ on sufficiently many random inputs, then (using the Hoeffiding bound) one can almost certainly distinguish the cases where $g(.)$ is $f(k,.)$ for a weak k and the case where $g(.)$ is a random oracle, as by definition of a weak key, the probability of A outputting 1 differs by at least $\epsilon'/2$ for both cases. Thus we can define an adversary which does the above sampling and outputs 0 and 1 respectively in the two above cases. As outlined, this adversary has a distinguishing advantage of at least $\epsilon'/2^{\lambda+1}$.[10] In the above argument it is important that in the case where $g(.)$ is a random oracle, we can sample many *independent* guess bits of A. This is not possible when considering "normal" PRFs, as then the adversary A can simply query $g(.)$ on some fixed inputs, and her guess bits will be completely dependent. This is the point in the proof where we exploit the fact that we consider *weak* PRFs.

1.3 Applications and Reductions

The unpredictability and indistinguishability based notions used in this paper are the strongest possible considering general leakage-functions, and a stream cipher satisfying them is sufficient to realize important primitives like stateful leakage-resilient symmetric authentication and encryption.[11]

It would be very interesting to construct a leakage-resilient pseudorandom function, as then we could implement those symmetric primitives in a *stateless* way. Let us mention here that cryptographic reductions, like the GGM construction of PRFs form PRGs [21], will in general *not* preserve leakage-resilience.

1.4 Notation

For a set $\mathcal{X}$, we denote with $X \xleftarrow{*} \mathcal{X}$ that X is assigned a value sampled uniformly at random from $\mathcal{X}$. To save on notation, we write X^i to denote a sequence $X_1, \ldots, X_i$. $\mathbf{R}_{n,m}$ denotes a uniformly random function $\{0,1\}^n \to \{0,1\}^m$, $\mathbf{R}_n$ denotes $\mathbf{R}_{n,n}$.

2 Leakage-Resilient Stream-Cipher from a Weak PRF

Figure 1 illustrates the mode of operation for which we prove that it gives a leakage-resilient stream cipher if instantiated with any weak PRF. Below we first

[10] The expression (3) in Lemma 2 is a bit more complicated than that. The last term in (3) is the error from the Hoeffding bound, and the second to last term is due to the fact that the sampled outputs are not completely independent as required by the Hoeffding bound.

[11] For authentication it is sufficient that the secret X_i used is unpredictable, thus here we can allow the adversary to observe the leakage in the round where X_i is computed. For semantically secure encryption, e.g. when using a one-time pad $C = M \oplus X_i$, we need X_i to be indistinguishable, thus here the adversary cannot get the leakage in round i, but can so for all other rounds $j < i$ (and, as we have forward security, also $j > i$).

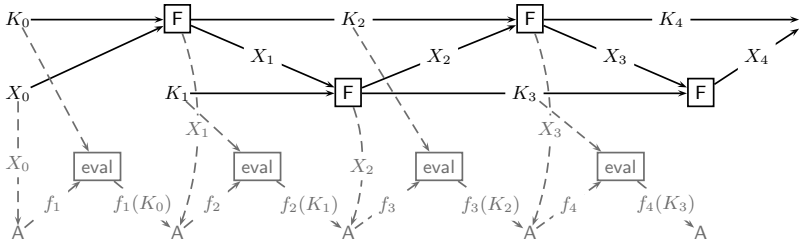


Fig. 1. Leakage resilient stream-cipher S^{F} from a seed-incompressible weak pseudorandom function F. The regular evaluation is shown in black, the attack related part is shown in gray with dashed lines.

define this construction, and state a Theorem which bounds the security of S^{F} as a *normal* stream-cipher. We then define what a *leakage-resilient* stream-cipher is. Then we state our main theorem (Theorem 2) which bounds the security of S^{F} as a leakage-resilient stream-cipher in terms of the security of F as a weak PRF.

The Construction: Let $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^{\kappa+n}$ be a function. Then, with S^{F} we denote the following simple construction of a stream cipher.

Initialization: The initial state is $S_0 = [K_0, K_1, X_0]$, where $K_0, K_1 \xleftarrow{*} \{0,1\}^\kappa$ and $X_0 \xleftarrow{*} \{0,1\}^n$. Only K_0, K_1 must be secret, X_0 can be public.
State: The state before the ith round is $S_{i-1} = [K_{i-1}, K_i, X_{i-1}]$.
Computation: In the ith round, $\mathsf{S}^{\mathsf{F}}(S_{i-1})$ computes

$$(K_{i+1}, X_i) := \mathsf{F}(K_{i-1}, X_{i-1})$$

and outputs X_i. Then the state $S_{i-1} = [K_{i-1}, K_i, X_{i-1}]$ is replaced with $S_i = [K_i, K_{i+1}, X_i]$ (note that K_i is not accessed in the ith round).

Security of S without Side-Channels: Theorem 1 below states that the output of S^{F} is pseudorandom (i.e. is a secure stream-cipher in the "classical" sense) if F is a secure weak pseudorandom function. The proof of this theorem is a straight forward hybrid argument and for space reasons only give in the full version of this paper [37]. The security of S^{F} is stated in terms of the security of F as a weak pseudorandom function (wPRF), which is defined like a normal PRF except that the inputs are random and not adversarially chosen.

Definition 1 (wPRF). $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^m$ *is a* <u>*weak*</u> $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$*-secure pseudorandom function (wPRF) if for all* A *of size* s_{prf} *and for random variables* $K \xleftarrow{*} \{0,1\}^\kappa$ *and for* $i = 1, \ldots, q_{\mathsf{prf}}$

$$X_i \xleftarrow{*} \{0,1\}^m \qquad Y_i = \mathsf{F}(K, X_i) \qquad R_i = \mathbf{R}_{n,m}(X_i)$$

we have $\Pr[\mathsf{A}(X^{q_{\mathsf{prf}}}, Y^{q_{\mathsf{prf}}}) = 1] - \Pr[\mathsf{A}(X^{q_{\mathsf{prf}}}, R^{q_{\mathsf{prf}}}) = 1] \leq \epsilon_{\mathsf{prf}}$

Theorem 1 (Security without Leakage). *If* F *is a* $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, 1)$ *secure wPRF, then for any* $\ell \in \mathbb{N}$*, no adversary of size* $s_{\mathsf{prf}} - \ell \cdot |\mathsf{F}|$ *can distinguish the first* $\ell+1$ *blocks as output by* S^{F} *from uniformly random with advantage more than* $\ell \cdot \epsilon_{\mathsf{prf}}$*.*

Side-Channel Adversary: As outlined in Section 1.1, we consider an adversary A who can attack S^{F} by choosing any function $f_i : \{0,1\}^\kappa \to \{0,1\}^\lambda$ before round i, and at the end of the round receives the normal output X_i of S^{F} and also the leakage $\Lambda_i \stackrel{\mathsf{def}}{=} f_i(K_{i-1})$. In round i, $\mathsf{S}^{\mathsf{F}}(S_{i-1})$ only access K_{i-1} and X_{i-1}, thus giving K_{i-1} as input to f_i means that f_i can use the entire state that S^{F} accesses in round i. Note that we don't have to explicitly give X_{i-1} as input to f_i, as A must only decide on f_i after she got X_{i-1} and thus can hard-code it into f_i. We denote with $\mathcal{A}_\lambda$ the set of adversaries as just described restricted to choose leakage functions with range $\{0,1\}^\lambda$.

Leakage-Resilient Security Notion: Let $\mathtt{view}_\ell$ denote the view of the adversary after X_ℓ has been computed, i.e.

$$\mathtt{view}_\ell = [X_0, \ldots, X_\ell, \Lambda_1, \ldots, \Lambda_\ell].$$

With $\mathtt{view}_\ell^- = \mathtt{view}_\ell \setminus X_\ell$ we denote $\mathtt{view}_\ell$ but without the last output X_ℓ. The security notion we consider requires that $X_{\ell+1}$ is *indistinguishable* from random, even when given $\mathtt{view}_\ell$ (which will imply that it is *unpredictable* given $\mathtt{view}_{\ell+1}^-$).

We denote with $\mathsf{S}(S_0) \stackrel{\ell}{\rightsquigarrow} \mathsf{A}$ the random experiment where an adversary $\mathsf{A} \in \mathcal{A}_\lambda$ attacks S (initialized with $S_0 = [K_0, K_1, X_0]$) for ℓ rounds (cf. Fig. 1), and with $\mathtt{view}(\mathsf{S}(S_0) \stackrel{\ell}{\rightsquigarrow} \mathsf{A})$ we denote the view $\mathtt{view}_\ell$ of A at the end of the attack. For any circuit $\mathsf{D} : \{0,1\}^* \to \{0,1\}$ (with one bit output), we denote with $\mathsf{AdvInd}(\mathsf{D}, \mathsf{A}, \mathsf{S}, \ell)$ the advantage of D in distinguishing K_ℓ from a random $U_n \stackrel{*}{\leftarrow} \{0,1\}^n$ given $\mathtt{view}(\mathsf{S}(S_0)S \stackrel{\ell-1}{\rightsquigarrow} \mathsf{A})$, formally

$$\mathsf{AdvInd}(\mathsf{D}, \mathsf{A}, \mathsf{S}, \ell) = |p_{real} - p_{rand}| \quad \text{where}$$

$$p_{rand} \stackrel{\mathsf{def}}{=} \Pr_{S_0}[\mathsf{D}(\mathtt{view}(\mathsf{S}(S_0) \stackrel{\ell-1}{\rightsquigarrow} \mathsf{A}), U_n) = 1]$$

$$p_{real} \stackrel{\mathsf{def}}{=} \Pr_{S_0}[\mathsf{D}(\mathtt{view}(\mathsf{S}(S_0) \stackrel{\ell-1}{\rightsquigarrow} \mathsf{A}), X_\ell) = 1]$$

Security of S against Side-Channel Attacks: The security of S^{F} will depend on the security of F as a weak pseudorandom function. Recall that the complexity a non-uniform adversary is captured by the size of a circuit describing it. For a circuit D, we let $\mathtt{size}(\mathsf{D})$ denote its size. We will also write $\mathtt{size}(\mathsf{S} \stackrel{\ell-1}{\rightsquigarrow} \mathsf{A})$ to denote the size of a circuit needed to implement the entire random experiment $\mathsf{S} \stackrel{\ell-1}{\rightsquigarrow} \mathsf{A}$, as illustrated in Figure 1, where eval denotes a circuit which on input the description of a function $f : \{0,1\}^\kappa \to \{0,1\}^\lambda$ and $K \in \{0,1\}^\kappa$ computes and outputs $f(K)$.

Theorem 2 (Security with Leakage). *Let* $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^{\kappa+n}$ *be a* $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, n/\epsilon_{\mathsf{prf}})$*-secure wPRF where* $\epsilon_{\mathsf{prf}} \geq n \cdot 2^{-n/3}$ *and* $n \geq 20$*. Let* $\lambda = \log(\epsilon_{\mathsf{prf}}^{-1})/6$ *and* $s' = s_{\mathsf{prf}}\epsilon_{\mathsf{prf}}^2/2^{\lambda+2}(n+\kappa)^3$*. Then for any adversary* $\mathsf{A} \in \mathcal{A}_\lambda$ *and distinguisher* D *where* $\mathtt{size}(\mathsf{S} \overset{\ell-1}{\rightsquigarrow} \mathsf{A}) + \mathtt{size}(\mathsf{D}) \leq s'$ *we have for any* $\ell \in \mathbb{N}$

$$\mathsf{AdvInd}(\mathsf{D}, \mathsf{A}, \mathsf{S}, \ell) \leq 8 \cdot \ell \cdot \epsilon_{\mathsf{prf}}^{1/12} \tag{1}$$

On λ: Note that the amount of leakage $\lambda = \log(\epsilon_{\mathsf{prf}}^{-1})/6$ we tolerate depends on the hardness of the underlying wPRF. Thus if F is secure against adversaries of super-polynomial size, i.e. $\epsilon_{\mathsf{prf}} = 2^{\omega(\log \kappa)}$, then the amount of leakage is at least super-logarithmic $\lambda = \omega(\log \kappa)$. This already covers many practical attacks like Hamming weight attacks (see e.g. [30]).

If F is exponentially hard, i.e. $\epsilon_{\mathsf{prf}} = 2^{-\Omega(\kappa)}$, then $\lambda = \Omega(\kappa)$, and thus we can even leak a constant fraction of the internal state in each round.

Security loss: The security loss in the theorem is significant: the 12th root of ϵ_{prf} comes up in the distinguishing advantage. In the full version [37] we discuss several approaches which potentially can be used to prove a much better bound.

Unpredictability: Theorem 2 states that when given the view of an adversary A who attacked S for $\ell - 1$ rounds, the next value X_ℓ to be computed is indistinguishable from random. If the adversary is also given $\Lambda_\ell = f_\ell(K_{\ell-1})$ (i.e. the leakage computed in round ℓ), X_ℓ cannot be pseudorandom any more, as Λ_ℓ could e.g. be the λ first bits of X_ℓ. In the case where Λ_ℓ is also leaked, one can still prove (using Lemma 4) that X_ℓ is unpredictable: for any $\delta > 0$, with probability $1-\delta$ the random variable X_ℓ has $n - \lambda - \log(\delta^{-1})$ bits of "HILL-pseudoentropy" (a notion to be defined in Section 4).

Forward Security: Like the construction from [17], also S^F is forward secure: Theorem 2 holds even for a stronger security notion than AdvInd, where the distinguisher D is additionally given entire state of S^F after round $\ell + 1$.

Instantiation with a block-cipher: Our construction requires a wPRF $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^{\kappa+n}$. Such an F can be constructed from any secure block-cipher $\mathsf{BC} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ like AES. (AES comes with different security parameters $\kappa = n = 128$ and $\kappa = n = 256$). For this we have to do some range expansion, e.g. by setting ($\|$ denotes concatenation)

$$\mathsf{F}(K, X) = \mathsf{BC}(K, X\|0))\|\mathsf{BC}(K, X\|1). \tag{2}$$

Here $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^{n-1} \to \{0,1\}^{2n}$ is a secure PRF (and thus wPRF) assuming that $\mathsf{BC} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is a pseudorandom permutation, which is the standard security notion for block-ciphers. [12]

[12] Let us stress, that just assuming that BC is a wPRF is not sufficient as (2) is *not* a secure range expansion of wPRFs (see e.g. [38] for some secure constructions).

3 wPRF with Non-uniform Keys and Inputs

We will need the following classical technical lemma several times.

Lemma 1 (Hoeffding's inequality [26]). *Let $X_1, \ldots, X_t$ be independent random variables where for $1 \leq i \leq t : \Pr(X_i \in [a_i, b_i]) = 1$. Then, for the sum of these variables $X = X_1 + \cdots + X_t$ we have the inequality:*

$$\Pr[X - \mathrm{E}[X] \geq t\epsilon] \leq \exp\left(-\frac{2\,t^2\,\epsilon^2}{\sum_{i=1}^{t}(b_i - a_i)^2}\right)$$

Recall that a random variable Z has min-entropy k, denoted $H_\infty(Z) = k$ if for all z in the range of Z we have $\Pr[Z = z] \leq 2^{-k}$.

Definition 2 (wPRF with non-uniform keys and inputs). *We call a function $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^m$ a $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$-secure wPRF with α-low keys, if it's a wPRF as in Definition 1, whenever the key K comes from any distribution with min-entropy $\kappa - \alpha$ (and not uniformly random).*

Similarly, we say F is a $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$-secure wPRF with β-low inputs, if it's a wPRF as in Definition 1, except that the inputs X_i come from any distribution with min-entropy $m - \beta$.

Non-uniform Keys. By the following lemma, every wPRF (using uniform keys) is a wPRF for α-low keys. The loss in security is roughly $2^{\alpha+1}$, which is almost optimal.

Lemma 2. *For any $\alpha > 0$ and $t \in \mathbb{N}$: If $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^m$ is a $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$-secure wPRF (for uniform keys), then it is a $(\epsilon'_{\mathsf{prf}}, s'_{\mathsf{prf}}, q'_{\mathsf{prf}})$-secure wPRF with α-low keys if the following holds*[13]

$$\begin{aligned} q_{\mathsf{prf}} &\geq q'_{\mathsf{prf}} \cdot t \\ \epsilon_{\mathsf{prf}} &\leq \epsilon'_{\mathsf{prf}}/2^{\alpha+1} - \frac{q_{\mathsf{prf}}^2}{2^{n+1}} - 2 \cdot \exp\left(-\frac{t^2 \cdot \epsilon'^2_{\mathsf{prf}}}{8}\right) \\ s_{\mathsf{prf}} &\geq s'_{\mathsf{prf}} \cdot t \end{aligned} \tag{3}$$

Proof. Assume there exists a random variable K_α where $H_\infty(K_\alpha) = \kappa - \alpha$, but where F is not a $(\epsilon'_{\mathsf{prf}}, s'_{\mathsf{prf}}, q'_{\mathsf{prf}})$-secure wPRF if the key is K_α. To prove the Lemma, we must show that then F is not $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$-secure wPRF for uniformly random keys. By assumption, there exists an adversary $\mathsf{A}, |\mathsf{A}| \leq s'_{\mathsf{prf}}$ such that

$$\sum_{k \in \{0,1\}^\kappa} \Pr[k = K_\alpha] \cdot \xi_k > \epsilon'_{\mathsf{prf}} \tag{4}$$

[13] As ϵ'_{prf} appears twice in eq.(3), we cannot easily express ϵ'_{prf} as a function of ϵ_{prf}. One can get a closed expression at the price of a worse bound by e.g. replacing ϵ'_{prf} in (3) with ϵ_{prf}, one then gets (for $t \in \mathbb{N}$ of our choice): $q'_{\mathsf{prf}} := q_{\mathsf{prf}}/t$, $s'_{\mathsf{prf}} := s_{\mathsf{prf}}/t$, $\epsilon'_{\mathsf{prf}} := 2^{\alpha+1}\left(\epsilon_{\mathsf{prf}} + q^2_{\mathsf{prf}}/2^{n+1} + 2 \cdot \exp\left(-t^2 \cdot \epsilon^2_{\mathsf{prf}}/8\right)\right)$.

where ξ_k denotes A's advantage conditioned on the key being k, i.e. with $X_i \xleftarrow{*} \{0,1\}^n$, $Y_i = \mathsf{F}(k, X_i)$, $R_i \leftarrow \mathbf{R}_{n,m}(X_i)$ (for $i = 1, \ldots, q'_{\mathsf{prf}}$)

$$\xi_k \stackrel{\mathsf{def}}{=} \Pr[\mathsf{A}(X^{q'_{\mathsf{prf}}}, Y^{q'_{\mathsf{prf}}}) = 1] - \Pr[\mathsf{A}(X^{q'_{\mathsf{prf}}}, R^{q'_{\mathsf{prf}}}) = 1]$$

We say $k \in \{0,1\}^\kappa$ is weak if $\xi_k \geq \epsilon'_{\mathsf{prf}}/2$, and let $\mathcal{K} \subset \{0,1\}^\kappa$ denote the set of all weak keys. From (4) we get by Markov

$$\Pr[K_\alpha \in \mathcal{K}] \geq \epsilon'_{\mathsf{prf}}/2.$$

Let K be uniform over $\{0,1\}^\kappa$. If we define an event $\mathcal{E}$ depending on K by $\Pr[\mathcal{E}|K = k] = \Pr[K_\alpha = k]/2^{\alpha-\kappa}$ it satisfies (see [37] for the proof)

$$\Pr[\mathcal{E}] = 2^{-\alpha} \quad \text{and} \quad \Pr[K = k|\mathcal{E}] = \Pr[K_\alpha = k]$$

With this we can lower bound the probability that the uniformly random key K is weak as

$$\Pr[K \in \mathcal{K}] \geq \Pr[\mathcal{E}] \Pr[K \in \mathcal{K}|\mathcal{E}] = \Pr[\mathcal{E}] \Pr[K_\alpha \in \mathcal{K}] = \frac{\Pr[K_\alpha \in \mathcal{K}]}{2^\alpha} \geq \frac{\epsilon'_{\mathsf{prf}}}{2^{\alpha+1}} \tag{5}$$

We will construct an adversary $\tilde{\mathsf{A}}$, where for $X_i \xleftarrow{*} \{0,1\}^n, Y_i = \mathsf{F}(k, X_i), R_i \leftarrow \mathbf{R}_{n,m}(X_i)$ the adversary $\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, R^{q_{\mathsf{prf}}})$ (where $q_{\mathsf{prf}} = q'_{\mathsf{prf}} \cdot t$) will almost always output 0, whereas $\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, Y^{q_{\mathsf{prf}}})$ will almost always output 1 if $k \in \mathcal{K}$. So $\tilde{\mathsf{A}}$ breaks the security of F as a weak PRF with advantage at least $\epsilon_{\mathsf{prf}} \approx \Pr[k \in \mathcal{K}] \geq \epsilon'_{\mathsf{prf}}/2^{\alpha+1}$. Let

$$\phi = \Pr[\mathsf{A}(X^{q_{\mathsf{prf}}}, R^{q_{\mathsf{prf}}}) = 1] \tag{6}$$

where the probability if over the choice of the $X_i \xleftarrow{*} \{0,1\}^n$, the random function $\mathbf{R}_{n,m}$ used to compute $R_i = \mathbf{R}_{n,m}(X_i)$ and also A (if it's not deterministic). Our adversary $\tilde{\mathsf{A}}$ on input $X^{q_{\mathsf{prf}}}, Z^{q_{\mathsf{prf}}}$, does the following.

- Split the input in t equal parts which we denote $(\hat{X}_1, \hat{Z}_1), \ldots, (\hat{X}_t, \hat{Z}_t)$ (so e.g. $\hat{X}_i = X_{(i-1)q'_{\mathsf{prf}}+1}, \ldots, X_{i \cdot q'_{\mathsf{prf}}}$).
- For $i = 1, \ldots, t$ compute $T_i \leftarrow \mathsf{A}(\hat{X}_i, \hat{Z}_i)$ and let

$$T := \sum_{i=1}^{t} T_i$$

  If $(T - t \cdot \phi) \leq t \cdot \epsilon'_{\mathsf{prf}}/4$ then $\tilde{\mathsf{A}}$ outputs 0, otherwise she outputs 1.

By the following two claims, $\tilde{\mathsf{A}}$ will almost never output 1 if the Z_i are random, but will output 1 with probability almost $\epsilon_{\mathsf{prf}}/2^{\alpha+1}$ if the the Z_i were computed by F.

Claim 1. *Let $X_i \xleftarrow{*} \{0,1\}^n$ and $R_i = \mathbf{R}_{n,m}(X_i)$, then*

$$\Pr[\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, R^{q_{\mathsf{prf}}}) = 1] \leq \exp\left(-\frac{t^2 \cdot \epsilon'^2_{\mathsf{prf}}}{8}\right) + \frac{q^2_{\mathsf{prf}}}{2^{n+1}}$$

Proof. By definition $\tilde{\mathsf{A}}$ will output 1 iff $(T - t \cdot \phi) > t \cdot \epsilon'_{\mathsf{prf}}/4$. In the case where the Z_i are computed as $\mathbf{R}_{n,m}(X_i)$ (as it is the case for the R_i in this claim) we have by eq.(6) $t \cdot \phi = \mathrm{E}[T]$, thus

$$\Pr[\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, R^{q_{\mathsf{prf}}}) = 1] = \Pr\left[T - \mathrm{E}[T] > t \cdot \frac{\epsilon'_{\mathsf{prf}}}{4}\right] \tag{7}$$

Let $T'_1, \ldots, T'_t$ be independent binary random variables, where for $j = 1, \ldots, t$ the T_j is sampled by choosing a uniformly random function $\mathbf{R}^j : \{0,1\}^n \to \{0,1\}^m$ and (for $i = 1, \ldots, q'_{\mathsf{prf}}$) $X_{j,i} \xleftarrow{*} \{0,1\}^n$, $R_{j,i} = \mathbf{R}^j(X_i)$ and setting $T'_j = \mathsf{A}(X_{j,1}, \ldots, X_{j,q'_{\mathsf{prf}}}, R_{j,1}, \ldots, R_{j,q'_{\mathsf{prf}}})$. Further let $T' := \sum_{j=1}^{t} T'_j$. As the T'_j's are independent, we can use Hoeffding's inequality (Lemma 1) to upper bound

$$\Pr\left[T' - \mathrm{E}[T'] > t \cdot \frac{\epsilon'_{\mathsf{prf}}}{4}\right] \leq \exp\left(-\frac{t^2 \cdot \epsilon'^2_{\mathsf{prf}}}{8}\right) \tag{8}$$

This bound does not apply to (7), as unlike the T'_j, the T_j are not completely independent, as we use *the same* random function $\mathbf{R}_{n,m}$ for each T_j. We will show that this is not a big problem if the domain is large enough, as conditioned on all the X_i's being different, the R_i's will have the same distribution in the computation of the T_j and T'_j; Let $\mathcal{E}$ denote the event, which holds if all the $q_{\mathsf{prf}} = q'_{\mathsf{prf}} \cdot t$ values $X_{j,i}$ (sampled to compute T or T') are pairwise distinct. As those values are all sampled independently and uniformly from $\{0,1\}^n$, by the birthday bound

$$\Pr[\neg\mathcal{E}] \leq \frac{q^2_{\mathsf{prf}}}{2^{n+1}} \tag{9}$$

Conditioned on $\mathcal{E}$, the distribution of the T_i's and T'_i (and thus of T and T') is identical, in particular

$$\Pr\left[T' - \mathrm{E}[T'] > t \cdot \frac{\epsilon'_{\mathsf{prf}}}{4} \middle| \mathcal{E}\right] = \Pr\left[T - \mathrm{E}[T] > t \cdot \frac{\epsilon'_{\mathsf{prf}}}{4} \middle| \mathcal{E}\right] \tag{10}$$

The claim now follows from (7)-(10). □

Claim 2. *Let* $K \xleftarrow{*} \{0,1\}^\kappa$ *and for* $i = 1, \ldots, q_{\mathsf{prf}}$: $X_i \xleftarrow{*} \{0,1\}^n$ *and* $Y_i = \mathsf{F}(K, X_i)$*, then*

$$\Pr[\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, Y^{q_{\mathsf{prf}}}) = 1] \geq \frac{\epsilon'_{\mathsf{prf}}}{2^{\alpha+1}}\left(1 - \exp\left(-\frac{t^2 \cdot \epsilon'^2_{\mathsf{prf}}}{8}\right)\right)$$

Proof. We have

$$\Pr[\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, Y^{q_{\mathsf{prf}}}) = 1] \geq \Pr[K \in \mathcal{K}] \cdot \Pr[\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, Y^{q_{\mathsf{prf}}}) = 1 | K \in \mathcal{K}] \tag{11}$$

By (5) we can lower bound the first term on the right side in (11) as

$$\Pr[K \in \mathcal{K}] \geq \epsilon'_{\mathsf{prf}}/2^{\alpha+1} \tag{12}$$

It remains to upper bound the second term. For this recall that $\tilde{\mathsf{A}}$ outputs 0 if $|T - t \cdot \phi| > t \cdot \epsilon'_{\mathsf{prf}}/4$, where $T = \sum_{j=1}^{t} T_j$ and each T_j is the output of $\mathsf{A}(X^{q'_{\mathsf{prf}}}, Y^{q'_{\mathsf{prf}}})$ where $Y_i = \mathsf{F}(K, X_i)$ (here the $X^{q'_{\mathsf{prf}}}$ are independent for each j but K is fixed). If $K \in \mathcal{K}$, then by definition of $\mathcal{K}$ we have $|\mathrm{E}[T_j] - \phi| \geq \epsilon'_{\mathsf{prf}}/2$, and thus $\tilde{\mathsf{A}}$ will only output 0, if the value of T is bounded away by at least $t \cdot \epsilon'_{\mathsf{prf}}/4$ from its expectation, again using the Hoeffding bound

$$\Pr[\tilde{\mathsf{A}}(X^{q_{\mathsf{prf}}}, Y^{q_{\mathsf{prf}}}) = 0 | K \in \mathcal{K}] = \Pr\left[T - \phi > t \cdot \frac{\epsilon'_{\mathsf{prf}}}{4}\right] \leq \exp\left(-\frac{t^2 \cdot \epsilon'^2_{\mathsf{prf}}}{8}\right)$$

The claim follows from this equation and (11),(12). □

The bound on $\tilde{\mathsf{A}}$'s advantage ϵ_{prf} as claimed in the lemma follows from the two claims above. The bound on the size s_{prf} and number of queries q_{prf} made by $\tilde{\mathsf{A}}$ follows directly from the definition of $\tilde{\mathsf{A}}$. ■

Non-uniform Inputs. We just showed that a wPRF stays secure even if the key is not uniform. In the full version of the paper we prove a similar result for the case where the *inputs* are not uniformly random. We only consider the case where the adversary gets a single input/output pair.

Lemma 3. *Let $\beta > 0$, then if $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^m$ is a $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, 1)$-secure wPRF (for uniform inputs), it's also a $(\epsilon'_{\mathsf{prf}}, s'_{\mathsf{prf}}, 1)$-secure wPRF for β-low entropy input, where for any $t \in \mathbb{N}$*

$$\epsilon_{\mathsf{prf}} \leq \epsilon'_{\mathsf{prf}}/2^{\beta+1} - 2 \cdot \exp\left(-\frac{2 \cdot t^2 \cdot \epsilon'^2_{\mathsf{prf}}}{64}\right)$$

$$s_{\mathsf{prf}} \geq s'_{\mathsf{prf}} \cdot 2t$$

4 Proof of Theorem 2

Proof Sketch. We will prove the security of S^{F} (cf. Figure 1) by proving that if the state X_{i-1}, K_{i-1} accessed in round i is independent and has HILL-pseudoentropy $n - 2\lambda$ and $\kappa - 2\lambda$, respectively, then also the output X_i, K_{i+1} has such a HILL-pseudoentropy given the leakage $\Lambda_i = f(X_{i-1}, K_{i-1})$ (Lemma 7). Though we unavoidably get some degradation in the "quality" of the pseudoentropy (in terms of ϵ, s in Definition 3 below), this degradation is only additive, and thus we can sum it up over all rounds.[14]

[14] This summation to bound the degradation in security is quite tedious. It might seem that one could get a much simpler proof using a hybrid argument, where for the jth hybrid one would simply replace the output in the first j rounds (having high HILL-pseudoentropy) with some (indistinguishable) output having high min-entropy. Unfortunately we can't make this intuition work, the reason is that high HILL-pseudoentropy only implies existence of an indistinguishable random variable with high min-entropy, but gives no means as to how to sample it. Thus it is not clear how to efficiently sample the hybrids just described.

Basic Definitions. We denote with $\delta^{\mathsf{D}}(X;Y)$ the advantage of a circuit D in distinguishing the random variables X, Y, i.e.: $\delta^{\mathsf{D}}(X;Y) \stackrel{\mathsf{def}}{=} |\Pr[\mathsf{D}(X)=1] - \Pr[\mathsf{D}(Y)=1]|$. With $\delta_s(X;Y)$ we denote $max_{\mathsf{D}}\delta^{\mathsf{D}}(X;Y)$ where the maximum is over all circuits D of size s.

Definition 3 (HILL-pseudoentropy[25, 2]). *We say X has* HILL *pseudoentropy k, denoted by $\mathbf{H}^{\mathsf{HILL}}_{\epsilon,s}(X) \geq k$, if there exists a distribution Y with min-entropy $\mathbf{H}_\infty(Y) = k$ where $\delta_s(X;Y) \leq \epsilon$.*

Definition 4 (PRG). *A function $\mathsf{prg} : \{0,1\}^n \to \{0,1\}^m$ is a (δ, s)-secure pseudorandom generator (PRG) if $\delta_s(\mathsf{prg}(U_n)\ ;\ U_m) \leq \delta$.*

Thus $\mathsf{prg}(Z)$ is indistinguishable from random if $Z \stackrel{*}{\leftarrow} \{0,1\}^n$. If some function $f(Z)$ of the seed is leaked, then $\mathsf{prg}(Z)$ will not look random any more, as e.g. $f(Z)$ could just output some bits of $\mathsf{prg}(Z)$. The following lemma states that if the range of f is not too big, then $\mathsf{prg}(Z)$ will still have high HILL-pseudoentropy.

Lemma 4 (Pseudoentropy of a PRG, [17]). *Let $\mathsf{prg} : \{0,1\}^n \to \{0,1\}^m$ and $f : \{0,1\}^n \to \{0,1\}^\lambda$ (where $1 \leq \lambda < n < m$) be any functions. If prg is a $(\epsilon_{\mathsf{prg}}, s_{\mathsf{prg}})$-secure pseudorandom-generator, then for any $\epsilon, \Delta > 0$ satisfying $\epsilon_{\mathsf{prg}} \leq \frac{\epsilon^2}{2^\lambda} - 2^{-\Delta}$, we have with $Z \stackrel{*}{\leftarrow} \{0,1\}^n$ and for any $\epsilon_{\mathsf{HILL}} > 0$*

$$\Pr_{Z \stackrel{*}{\leftarrow} \{0,1\}^n}[\mathbf{H}^{\mathsf{HILL}}_{\epsilon+\epsilon_{\mathsf{HILL}},\hat{s}}(\mathsf{prg}(Z)|f(Z)) \geq m - \Delta] \geq 1 - \epsilon \tag{13}$$

where $\hat{s} \approx \epsilon^2_{\mathsf{HILL}} s_{\mathsf{prg}}/8m$.

We will use the following technical lemma about some general random processes to show that the inputs X_i and keys K_i in the computation of S^{F} are independent.

Lemma 5 ([16]). *Let A_0, B_0 be independent random variables, and $\phi_1, \phi_2, \ldots$ be any sequence of functions. Let $A_1, A_2, \ldots$, $B_1, B_2, \ldots$ and $V_1, V_2, \ldots$ be defined as*

$$\begin{gathered}((A_{i+1}, V_{i+1}), B_{i+1}) := (\phi_{i+1}(A_i, V_1, \ldots, V_i), B_i) \\ \textit{if } i \textit{ is even} \\ (A_{i+1}, (V_{i+1}, B_{i+1})) := (A_i, \phi_{i+1}(B_i, V_1, \ldots, V_i)) \\ \textit{otherwise}\end{gathered}$$

Then $B_i \to \{V_1, \ldots, V_i\} \to A_i$ (and $A_i \to \{V_1, \ldots, V_i\} \to B_i$) is a Markov chain (or equivalently, A_i and B_i are independent given the $V_1, \ldots, V_i$)

Combining Lemmata 2, 3 and 4, we can prove Lemma 6 below, which states that the output $\mathsf{F}(K, X)$ of a wPRF has high HILL-pseudoentropy, even if K and X have high min-entropy (but are independent) and given some leakage $f(K, X)$. We set $t = n/\epsilon_{\mathsf{prf}}$ in Lemma 2 and 3, moreover we need the domain $\{0,1\}^n$ of F to be large enough, in particular, we will assume that (with ϵ_{prf} as in the lemma below)

$$\epsilon_{\mathsf{prf}} \geq \frac{n^2}{2^{n+1} \cdot \epsilon^2_{\mathsf{prf}}} + 2\exp(-n^2/32) \tag{14}$$

Note that the term on the right side drops exponentially in n, thus this restriction is a very weak one, and is e.g. satisfied for any $\epsilon_{\mathsf{prf}} \geq n \cdot 2^{-n/3}$ and $n \geq 20$.

Lemma 6. *Let* $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^m$ *be a* $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, n/\epsilon_{\mathsf{prf}})$*-secure wPRF. Let* $K \in \{0,1\}^\kappa$ *and* $X \in \{0,1\}^n$ *be independent where* $H_\infty(K) = \kappa - 2\lambda$ *and* $H_\infty(X) = n - 2\lambda$ *and let* $f : \{0,1\}^{\kappa+n} \to \{0,1\}^\lambda$ *be any leakage function, then for large enough* n *(as just described) and* $\lambda \leq \log(\epsilon_{\mathsf{prf}}^{-1})/6$

$$\Pr_{X,Y}[\mathbf{H}^{\mathsf{HILL}}_{\epsilon',s'}(\mathsf{F}(K,X)|X, f(K,X)) \geq m - 2\lambda] \geq 1 - 2^{-\lambda/2+1}$$

with $\epsilon' = 2^{-\lambda/2+2}$ *and* $s' = s_{\mathsf{prf}}/2^{\lambda+3}(n+\kappa)^3$.

Proof. We set $\Delta := 2\lambda$ and $\epsilon = \epsilon_{\mathsf{HILL}} := 2^{-\lambda/2+1}$, and require that

$$\lambda \leq 2 + \log(\epsilon_{\mathsf{prg}}^{-1})/2 \tag{15}$$

so that it satisfies the condition $\epsilon_{\mathsf{prg}} \leq \frac{\epsilon^2}{2^\lambda} - 2^{-\Delta}$ from Lemma 4, where now we can write (13) as

$$\Pr_{Z \xleftarrow{*} \{0,1\}^n}[\mathbf{H}^{\mathsf{HILL}}_{2^{-\lambda/2+2},\hat{s}}(\mathsf{prg}(Z)|f(Z)) \geq m - 2\lambda] \geq 1 - 2^{-\lambda/2+1} \tag{16}$$

where $\hat{s} = s_{\mathsf{prg}}/2^{\lambda+1}(n+\kappa)$. Now consider the wPRF F from the statement of the lemma, first we apply Lemma 2 with $t = n/\epsilon_{\mathsf{prf}}$ and $q_{\mathsf{prf}} = t$ to get for a uniformly random X' (in the second step below we use eq.(14)).

$$\delta_{s_{\mathsf{prf}}\epsilon_{\mathsf{prf}}/n}(\mathsf{F}(K,X')\|X' \ ; \ U_m\|X') \leq$$
$$\epsilon_{\mathsf{prf}} \cdot 2^{\Delta+1} + 2^{\Delta+1}\left(n^2/2^{n+1} \cdot \epsilon_{\mathsf{prf}}^2 + 2\exp\left(-n^2/8\right)\right) \leq \epsilon_{\mathsf{prf}} \cdot 2^{\Delta+2}$$

Thus F is a $(s_{\mathsf{prf}}\epsilon_{\mathsf{prf}}/n, \epsilon_{\mathsf{prf}} \cdot 2^{\Delta+1}, 1)$ secure wPRF even if we use a non-uniform key K. Now we apply Lemma 3 (again with $t = n/\epsilon_{\mathsf{prf}}$ and using eq.(14) in the second step)

$$\delta_{s_{\mathsf{prf}}\epsilon_{\mathsf{prf}}^2/2n^2}(\mathsf{F}(K,X)\|X \ ; \ U_m\|X) \leq$$
$$\epsilon_{\mathsf{prf}} \cdot 2^{2\Delta+3} + 2^{\Delta+1} \cdot 2 \cdot \exp(-n^2/32) \leq \epsilon_{\mathsf{prf}} \cdot 2^{2\Delta+4}$$

Thus we can see F on input K, X as an $(\epsilon_{\mathsf{prg}}, s_{\mathsf{prg}})$-secure pseudorandom generator where $s_{\mathsf{prg}} = s_{\mathsf{prf}}\epsilon_{\mathsf{prf}}^2/2n^2$ and $\epsilon_{\mathsf{prg}} = \epsilon_{\mathsf{prf}} \cdot 2^{2\Delta+4}$ (note that eq.(15) is still satisfied as in the statement of the lemma we require $\lambda \leq \log(\epsilon_{\mathsf{prf}}^{-1})/6$).

Now consider any function $f : \{0,1\}^{\kappa+n} \to \{0,1\}^\lambda$, by (16)

$$\Pr_{K,X}[\mathbf{H}^{\mathsf{HILL}}_{\epsilon',s'}(\mathsf{F}(K,X)|f(K,X), X) \geq m - 2\lambda] \geq 1 - 2^{-\lambda/2+1}$$

with $\epsilon' = 2^{-\lambda/2+2}$ and $s' = s_{\mathsf{prg}}/2^{\lambda+1}(n+\kappa) > s_{\mathsf{prf}}\epsilon_{\mathsf{prf}}^2/2^{\lambda+2}(n+\kappa)^3$. ■

The following lemma quantifies the security loss in one round of our stream cipher. Let $\mathtt{size}_i$ denote the size of the circuit realizing the ith round of the experiment $\mathsf{S}^{\mathsf{F}} \overset{\ell}{\rightsquigarrow} \mathsf{A}$, then $\sum_{i=1}^{\ell} \mathtt{size}_i = \mathtt{size}(\mathsf{S} \overset{\ell}{\rightsquigarrow} \mathsf{A})$.

Lemma 7 (The ith round). *Consider the random experiment* $\mathsf{S}^{\mathsf{F}} \overset{\ell}{\rightsquigarrow} \mathsf{A}$. *Then if before round* $i \leq \ell$ *for some* $s_{i-1} \leq s'$ *(with* s', ϵ', λ *are as in the previous lemma)*

$$\mathbf{H}^{\mathsf{HILL}}_{\epsilon_{i-1}, s_{i-1}}(K_{i-1} | \mathtt{view}_{i-1}) \geq \kappa - 2\lambda$$
$$\mathbf{H}^{\mathsf{HILL}}_{\epsilon_{i-1}, s_{i-1}}(X_{i-1} | \mathtt{view}^{-}_{i-1}) \geq n - 2\lambda$$

then with probability $1 - 2^{-\lambda/2+1}$ *the output* $(K_{i+1}, X_i) = \mathsf{F}(K_{i-1}, X_{i-1})$ *satisfies*

$$\mathbf{H}^{\mathsf{HILL}}_{\epsilon_i, s_i}(\mathsf{F}(K_{i-1}, X_{i-1}) | \mathtt{view}^{-}_{i}) \geq \kappa + n - 2\lambda$$

where $\epsilon_i = \epsilon_{i-1} + \epsilon'$, $s_i = s_{i-1} + \mathtt{size}_i$

Proof. Consider random variables K'_{i-1}, X'_{i-1} which have high *min-entropy*

$$H_\infty(K'_{i-1} | \mathtt{view}_{i-1}) \geq \kappa - \lambda \quad \text{and} \quad H_\infty(X'_{i-1} | \mathtt{view}^{-}_{i-1}) \geq n - \lambda$$

By Lemma 6 with probability at least $1 - 2^{-\lambda/2+1}$

$$\mathbf{H}^{\mathsf{HILL}}_{\epsilon', s'}(\mathsf{F}(K'_{i-1}, X'_{i-1}) | \mathtt{view}^{-}_{i}) \geq \kappa + n - 2\lambda$$

holds with $\epsilon' = 2^{-\lambda/2+2}$ and $s' = \frac{s_{\mathsf{prf}}}{2^{\lambda+3} \cdot (n+\kappa)^3}$. If we now use the random variables K_{i-1}, X_{i-1} (only having high HILL-pseudoentropy) instead of K'_{i-1}, X'_{i-1}, we get (recall that $s_{i-1} < s'$)

$$\mathbf{H}^{\mathsf{HILL}}_{\epsilon' + \epsilon_{i-1}, s_{i-1} - \mathtt{size}_i}(\mathsf{F}(K_{i-1}, X_{i-1}) | \mathtt{view}^{-}_{i}) \geq \kappa + n - 2\lambda$$

Let us stress that here the new error ϵ_i is $\epsilon' + \epsilon_{i-1}$, and not $\epsilon' + 2\epsilon_{i-1}$, as one would think because we must add an error term of ϵ_{i-1} for K_{i-1} and X_{i-1} respectively. Such a weaker bound would render the lemma useless, as then ϵ_i would grow exponentially in i. The reason we only have to add ϵ_{i-1}, is that in round $i-1$, F outputs (X_{i-1}, K_i), and it's this tuple that cannot be distinguished with advantage more than ϵ_{i-1}. Thus by adding an error ϵ_{i-1} for X_{i-1} in round i, we also account for K_i to be used in the next round, and we won't have to add an extra error term there. ■

The bound on the security of S^{F} as stated in Theorem 2 now follows by summing up the security decrease in each round as stated in the previous lemma. To apply the lemma, one must show that for each i, the K_i and X_i are independent given the view of the adversary, this follows from Lemma 5 by identifying A_i (from the Lemma) with $K_{2(i-1)}$ (as computed by S^{F}), identifying B_i with $K_{2(i-1)+1}$ and V_i with $\mathtt{view}_i$. In particular, after ℓ round, the error adds up to

$$\mathsf{AdvInd}(\mathsf{D}, \mathsf{A}, \mathsf{S}, \ell) \leq \ell \cdot 2^{-\lambda/2+3}.$$

Note that this is a bit strange, as the advantage *decreases* by increasing the leakage λ, but this is only due to the fact that we explicitly set the error parameters ϵ and ϵ_{HILL} as functions of λ in the proof of Lemma 6 in order to keep the number of parameters down. Setting $\lambda = \log(\epsilon^{-1}_{\mathsf{prf}})/6$ (note that this is the largest value allowed in the statement of Lemma 6), we get the bound as claimed in the theorem.

Acknowledgements

I'd like to thank Stefan Dziembowski and Sebastian Faust for some interesting discussions and the authors of [36, 41] for clarifying their work.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: TCC (2009)
2. Barak, B., Shaltiel, R., Wigderson, A.: Computational analogues of entropy. In: RANDOM-APPROX, pp. 200–215 (2003)
3. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
5. Canetti, R., Eiger, D., Goldwasser, S., Lim, D.-Y.: How to protect yourself without perfect shredding. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 511–523. Springer, Heidelberg (2008)
6. Cash, D.M., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 479–498. Springer, Heidelberg (2007)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
8. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, p. 292. Springer, Heidelberg (1999)
9. Courtois, N.T., Bard, G.V., Wagner, D.: Algebraic and slide attacks on keeLoq. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 97–115. Springer, Heidelberg (2008)
10. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)
11. Dodis, Y., Sahai, A., Smith, A.: On perfect and adaptive security in exposure-resilient cryptography. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 301–324. Springer, Heidelberg (2001)
12. Dodis, Y., Wichs, D.: One-round authenticated key agreement from weak secrets. Cryptology ePrint Archive, Report 2008/503 (2008), http://eprint.iacr.org/
13. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
14. Dziembowski, S.: On forward-secure storage (extended abstract). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 251–270. Springer, Heidelberg (2006)
15. Dziembowski, S., Maurer, U.M.: On generating the initial key in the bounded-storage model. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 126–137. Springer, Heidelberg (2004)

16. Dziembowski, S., Pietrzak, K.: Intrusion-resilient secret sharing. In: FOCS, pp. 227–237 (2007)
17. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS (2008)
18. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.T.M.: On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 203–220. Springer, Heidelberg (2008)
19. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: CHES, pp. 251–261 (2001)
20. Goldreich, O.: A uniform-complexity treatment of encryption and zero-knowledge. Journal of Cryptology 6(1), 21–53 (1993)
21. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. In: FOCS, pp. 464–479 (1984)
22. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
23. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX Security Symposium, pp. 45–60 (2008)
24. Halevi, S., Myers, S., Rackoff, C.: On seed-incompressible functions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 19–36. Springer, Heidelberg (2008)
25. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
26. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(301), 13–30 (1963)
27. Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: A practical attack on KeeLoq. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 1–18. Springer, Heidelberg (2001)
28. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
29. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
30. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Side channel cryptanalysis of product ciphers. In: Quisquater, J.-J., Deswarte, Y., Meadows, C., Gollmann, D. (eds.) ESORICS 1998. LNCS, vol. 1485, pp. 97–110. Springer, Heidelberg (1998)
31. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
32. Kocher, P.C.: Design and validation strategies for obtaining assurance in countermeasures to power analysis and related attacks. In: Proceedings of the NIST Physical Security Workshop (2005)
33. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
34. Maurer, U.M.: A provably-secure strongly-randomized cipher. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 361–373. Springer, Heidelberg (1991)
35. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)

36. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T., Yung, M.: A block cipher based pseudo random number generator secure against side-channel key recovery. In: ASIACCS, pp. 56–65 (2008)
37. Pietrzak, K.: Full version of this paper, http://homepages.cwi.nl/ pietrzak/publications.html
38. Pietrzak, K., Sjödin, J.: Range extension for weak pRFs; the good, the bad, and the ugly. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 517–533. Springer, Heidelberg (2007)
39. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: E-smart, pp. 200–210 (2001)
40. Reingold, O., Trevisan, L., Tulsiani, M., Vadhan, S.P.: Dense subsets of pseudo-random sets. In: FOCS, pp. 76–85 (2008)
41. Standaert, F.-X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
42. Vadhan, S.P.: Constructing locally computable extractors and cryptosystems in the bounded-storage model. Journal of Cryptology 17(1), 43–77 (2004)

ECM on Graphics Cards*

Daniel J. Bernstein[1], Tien-Ren Chen[2], Chen-Mou Cheng[3], Tanja Lange[4], and Bo-Yin Yang[2]

[1] Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607–7045, USA
djb@cr.yp.to
[2] Institute of Information Science, Academia Sinica, 128 Section 2 Academia Road, Taipei 115-29, Taiwan, {by,trchen1033}@crypto.tw
[3] Department of Electrical Engineering, National Taiwan University, 1 Section 4 Roosevelt Road, Taipei 106-70, Taiwan, doug@crypto.tw
[4] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
tanja@hyperelliptic.org

Abstract. This paper reports record-setting performance for the elliptic-curve method of integer factorization: for example, 926.11 curves/second for ECM stage 1 with $B_1 = 8192$ for 280-bit integers on a single PC. The state-of-the-art GMP-ECM software handles 124.71 curves/second for ECM stage 1 with $B_1 = 8192$ for 280-bit integers using all four cores of a 2.4 GHz Core 2 Quad Q6600.

The extra speed takes advantage of extra hardware, specifically two NVIDIA GTX 295 graphics cards, using a new ECM implementation introduced in this paper. Our implementation uses Edwards curves, relies on new parallel addition formulas, and is carefully tuned for the highly parallel GPU architecture. On a single GTX 295 the implementation performs 41.88 million modular multiplications per second for a general 280-bit modulus. GMP-ECM, using all four cores of a Q6600, performs 13.03 million modular multiplications per second.

This paper also reports speeds on other graphics processors: for example, 2414 280-bit elliptic-curve scalar multiplications per second on an older NVIDIA 8800 GTS (G80), again for a general 280-bit modulus. For comparison, the CHES 2008 paper "Exploiting the Power of GPUs for Asymmetric Cryptography" reported 1412 elliptic-curve scalar multiplications per second on the same graphics processor despite having fewer bits in the scalar (224 instead of 280), fewer bits in the modulus (224 instead of 280), and a *special* modulus $(2^{224} - 2^{96} + 1)$.

Keywords: Factorization, graphics processing unit, modular arithmetic, elliptic curves, elliptic-curve method of factorization, Edwards curves.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* Permanent ID of this document: 6904068c52463d70486c9c68ba045839. Date of this document: 2009.01.26. This work was sponsored in part by the National Science Foundation under grant ITR–0716498, in part by Taiwan's National Science Council (grants NSC-96-2221-E-001-031-MY3 and -96-2218-E-001-001, also through the Taiwan Information Security Center NSC-97-2219-E-001-001, -96-2219-E-011-008), and in part by the European Commission through the ICT Programme under Contract ICT–2007–216676 ECRYPT II. Part of this work was carried out while Bernstein and Lange visited NTU.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 483–501, 2009.

1 Introduction

The elliptic-curve method (ECM) of factorization was introduced by Lenstra in [34] as a generalization of Pollard's $p-1$ and Williams' $p+1$ method. Many speedups and good choices of elliptic curves were suggested and ECM is now the method of choice to find factors in the range 10^{10} to 10^{60} of general numbers. The largest factor found by ECM was a 222-bit factor of the 1266-bit number $10^{381}+1$ found by Dodson (see [49]).

Cryptographic applications such as RSA use "hard" integers with much larger prime factors. The number-field sieve (NFS) is today's champion method of finding those prime factors. It was used, for example, in the following factorizations:

integer	bits	details reported
RSA–130	430	at ASIACRYPT 1996 by Cowie et al. [16]
RSA–140	463	at ASIACRYPT 1999 by Cavallar et al. [12]
RSA–155	512	at EUROCRYPT 2000 by Cavallar et al. [13]
RSA–200	663	in 2005 posting by Bahr et al. [4]
$2^{1039}-1$	1039 (special)	at ASIACRYPT 2007 by Aoki et al. [2]

A 1024-bit RSA factorization by NFS would be considerably more difficult than the factorization of the special integer $2^{1039}-1$ but has been estimated to be doable in a year of computation using standard PCs that cost roughly $1 billion or using ASICs that cost considerably less. See [43], [35], [19], [22], [44], and [29] for various estimates of the cost of NFS hardware. Current recommendations for RSA key sizes — 2048 bits or even larger — are based directly on extrapolations of the speed of NFS.

NFS is also today's champion index-calculus method of computing discrete logarithms in large prime fields, quadratic extensions of large prime fields, etc. See, e.g., [26], [27], and [5]. Attackers can break "pairing-friendly elliptic curves" if they can compute discrete logarithms in the corresponding "embedding fields"; current recommendations for "embedding degrees" in pairing-based cryptography are again based on extrapolations of the speed of NFS. See, e.g., [30].

NFS factors a "hard" integer n by combining factorizations of many smaller auxiliary "smooth" integers. For example, the factorization of RSA-155 $\approx 2^{512}$ generated a pool of $\approx 2^{50}$ auxiliary integers $< 2^{200}$, found $\approx 2^{27}$ "smooth" integers factoring into primes $< 2^{30}$, and combined those integers into a factorization of RSA-155. See [13] for many more details.

Textbook descriptions of NFS state that prime factors of the auxiliary integers are efficiently discovered by sieving. However, sieving requires increasingly intolerable amounts of memory as n grows. Cutting-edge NFS computations control their memory consumption by using other methods — primarily ECM — to discover large prime factors. Unlike sieving, ECM remains productive with limited amounts of memory.

Aoki et al. in [2] discovered small prime factors by sieving, discarded any unfactored parts above 2^{105}, and then used ECM to discover primes up to 2^{38}. Kleinjung reported in [29, Section 5] on ECM "cofactorisation" for a 1024-bit n consuming, overall, a similar amount of time to sieving.

The size of the auxiliary numbers to be factored by ECM depends on the size of the number to be factored with the NFS and on the relative speed of the ECM implementation. The SHARK design [19] for factoring 1024-bit RSA makes two suggestions for parameters of ECM — one uses it for 125-bit numbers, the other for 163-bit numbers. The SHARK designers remark that ECM could be used more intensively. In their design, ECM can be handled by conventional PCs or special hardware. They write "Special hardware for ECM ... can save up to 50% of the costs for SHARK" and "The importance of using special hardware for factoring the potential sieving reports grows with the bit length of the number to be factored." As a proof of concept Pelzl et al. present in [40] an FPGA-based implementation of ECM for numbers up to 200 bits and state "We show that massive parallel and cost-efficient ECM hardware engines can improve the area-time product of the RSA moduli factorization via the GNFS considerably." Gaj et al. [20] consider the same task and improve upon their results.

Evidently ECM is becoming one of the most important steps in the entire NFS computation. Speedups in ECM are becoming increasingly valuable as tools to speed up NFS.

This paper suggests graphics processing units (GPUs) as computation platforms for ECM, presents algorithmic improvements that are particularly helpful in the GPU context, and reports new ECM implementations for several NVIDIA GPUs. GPUs achieve high throughput through massive parallelism — usually more than 100 "cores" running at clock frequencies not much lower than that of state-of-the-art CPUs; e.g., the NVIDIA GeForce 8800 GTS 512 has 128 cores running at 1.625 GHz. This parallelism is well suited for ECM factorizations inside the NFS, although it also creates new resource-allocation challenges, as discussed later in this paper. We focus on moduli of 200–300 bits since we (correctly) predicted that our ECM implementation would be faster than previous ones and since we are looking ahead to larger NFS factorizations than 1024 bits.

Measurements show that a computer running this paper's new ECM implementation on a GPU performs 41.88 million 280-bit modular multiplications per second and has a significantly better price-performance ratio than a computer running the state-of-the-art GMP-ECM software on all four cores of a Core 2 Quad CPU. The best price-performance ratio is obtained by a computer that has a CPU and two GPUs contributing to the ECM computation.

2 Background on ECM

A thorough presentation of ECM is given by Zimmermann and Dodson in [48]. Their paper also describes extensive details of the GMP-ECM software, essentially

the fastest known ECM implementation to date. For more recent improvements of bringing together ECM with the algorithmic advantages of Edwards curves and improved curve choices we refer to [8] by Bernstein et al.

2.1 Overview of ECM

ECM tries to factor an integer m as follows.

Let E be an elliptic curve over $\mathbf{Q}$ with neutral element O. Let P be a non-torsion point on E. If the discriminant of the curve or any of the denominators in the coefficients of E or P happens not to be coprime with m without being divisible by it we have found a factor and thus completed the task of finding a nontrivial factor of m; if one of them is divisible by m we choose a different pair (E, P). We may therefore assume that E has good reduction modulo m. In particular we can use the addition law on E to define an addition law on $\tilde{E}$, the reduction of E modulo m; let $\tilde{P} \in \tilde{E}$ be the reduction of P modulo m.

Let ϕ be a rational function on E which has a zero at O and has non-zero reduction of $\phi(P)$ modulo m. In the familiar case of Weierstrass curves this function can simply be Z/Y. For elliptic curves in Edwards form a similarly simple function exists; see below.

Let s be an integer that has many small factors. A standard choice is $s = \mathrm{lcm}(1, 2, 3, \ldots, B_1)$. Here B_1 is a bound controlling the amount of time spent on ECM. The main step in ECM is to compute $R = [s]\tilde{P}$. The computation of the scalar multiple $[s]\tilde{P}$ on $\tilde{E}$ is done using the addition law on E and reducing intermediate results modulo m.

One then checks $\gcd(\phi(R), m)$; ECM succeeds if the gcd is nontrivial. If this first step — called stage 1 — was not successful then one enters stage 2, a postprocessing step that significantly increases the chance of factoring m. In a simple form of stage 2 one computes $R_1 = [p_{k+1}]R, R_2 = [p_{k+2}]R, \ldots, R_\ell = [p_{k+\ell}]R$ where $p_{k+1}, p_{k+2}, \ldots, p_{k+\ell}$ are the primes between B_1 and another bound B_2, and then does another gcd computation $\gcd(\phi(R_1)\phi(R_2)\cdots\phi(R_\ell), m)$. There are more effective versions of stage 2. Stage 2 takes significantly less time than stage 1 when ECM as a whole is optimized.

If q is a prime divisor of m, and the order of P modulo q divides s, then $\phi([s]\tilde{P}) \equiv 0 \pmod{q}$. If $\phi([s]\tilde{P}) \not\equiv 0 \bmod m$ we obtain a nontrivial factor of m in stage 1 of ECM as $\gcd(m, \phi([s]\tilde{P}))$. This happens exactly if there are two prime divisors of m such that s is divisible by the order of P modulo one of them but not modulo the other. Choosing s to have many small factors increases the chance of m having at least one prime divisor q such that the order of P modulo q divides s. Note that it is rare that this happens for all factors of m simultaneously unless s is huge.

Similar comments apply to stage 2, with s replaced by sp_{k+1}, sp_{k+2}, etc.

Trying a single curve with a large B_1 is usually less effective than spending the same amount of time trying many curves, each with a smaller B_1. For each curve one performs stage 1 and then stage 2.

2.2 Edwards Curves

Edwards curves were introduced by Edwards in [18] and studied for cryptography by Bernstein and Lange in [10]. An Edwards curve is given by an equation of the form $x^2 + y^2 = 1 + dx^2y^2$, for some $d \notin \{0, 1\}$. Bernstein and Lange showed that each elliptic curve with a point of order 4 is birationally equivalent to an Edwards curve over the same field. For ECM we are interested in curves with smooth order modulo factors of m, so in particular the condition of having a point of order 4 is not a problem. On the contrary, curves with large $\mathbf{Q}$-rational torsion subgroup are more likely to lead to factorizations since the torsion subgroup is mapped injectively under reduction. For our implementation we used Edwards curves with $\mathbf{Q}$-torsion group isomorphic to $\mathbf{Z}/2 \times \mathbf{Z}/8$ which were generated with the Edwards analogue of the Atkin-Morain construction [3] as described in [8].

The addition law on Edwards curves is given by

$$(x_1, y_1) \oplus (x_2, y_2) = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right).$$

The neutral element is $(0, 1)$, so ϕ in the previous subsection can simply be the x-coordinate.

For an overview of explicit formulas for arithmetic on elliptic curves we refer to the Explicit-Formulas Database (EFD) [9]. For doublings on Edwards curves we use the formulas by Bernstein and Lange [10]. For additions we use the mixed-addition formulas by Hisil et al. [25, page 8] to minimize the number of multiplications. Earlier formulas by Bernstein and Lange are complete, and can take advantage of fast multiplications by small parameters d, but completeness is not helpful for ECM, and curves constructed by the Atkin-Morain method have large d.

Note that the paper [8] also contains improvements of ECM using curves with small coefficients and base point, in which case using inverted twisted Edwards coordinates (see [7]) becomes advantageous. In Section 4 we describe how we implemented modular arithmetic on the GPU. The chosen representation does not give any speedup for multiplication by small parameters. This means that we do not get the full benefit of [8] — but we still benefit from the faster elliptic-curve arithmetic and the more successful curve choices on top of the fast and highly parallel computation platform. In Section 5 we present new parallelized formulas for Edwards-curve arithmetic.

3 Review of GPUs and GPU Programming

Today's graphics cards contain powerful GPUs to handle the increasing complexity and screen resolution in video games. GPUs have now developed into a powerful, highly parallel computing platform that finds more and more interest outside graphics-processing applications. In cryptography so far mostly secret-key applications were implemented (see, e.g., [14] and the book [15]) while taking full advantage of GPUs for public-key cryptography remained a challenge [38].

Along with the G80 series of GPUs, NVIDIA introduced CUDA, a parallel computing framework with a C-like programming language specifically intended for compilation and execution on a GPU. In this section we describe current NVIDIA graphics cards used for our implementations, give some background information on CUDA programming, and compare NVIDIA GPUs to AMD GPUs.

3.1 The NVIDIA Cards Used for CUDA

An NVIDIA GPU contains many streaming multiprocessors (MPs), each of which contains the following elements:

- a scheduling and dispatching unit that can handle many lightweight threads;
- eight (8) "cores" (often called streaming processors, or SPs) each capable of a fused single-precision floating-point multiply-and-add (MAD), or otherwise one 32-bit integer add/subtract or logical operation every cycle;
- two (2) "super function units" that each can do various complex computations like 32-bit integer multiplications, floating-point divisions, or *two (2) single-precision floating-point multiplications per cycle*;
- *for the more advanced GT2xx GPUs*, additional circuitry that in conjunction with the SPs can do double-precision floating-point arithmetic, albeit with a lower throughput (roughly 1/6 of that of single-precision counterpart);
- fast local shared memory, 16 banks of 1 kB each;
- controllers to access uncached thread-local and global memory;
- fast local read-only cache to device memory on the card, up to 8 kB;
- fast local read-only cache on, and access to, a *texture unit* (2 MPs on a G8x or G9x, and 3 MPs on a GT2xx form a cluster sharing a texture unit);
- a file of 8192 (for G8x or G9x) or 16384 (for GT2xx) 32-bit registers.

Uncached memory has a relatively low throughput and long latency. For example, the 128 SPs on a GeForce 8800 GTX run at 1.35 GHz, and the uncached memory provides a throughput of 86.4 GB/s. That may sound impressive but it is only a single 32-bit floating-point number per cycle per MP, with a latency of 400–600 cycles to boot.

The GPU can achieve more impressive data movement by broadcasting the same data to many threads in a cycle. The shared memory in an MP can deliver 64 bytes every two cycles, or 4 bytes per cycle per SP *if there is no bank conflict*. Latencies of all caches and shared memories are close to that of registers and hence much lower than device memories.

G8x/G9x Series. The chip used in the GeForce 8800 GTX is a typical NVIDIA G80-series GPU, a 90nm-process GPU containing 128 SPs grouped into 16 MPs.

G92 GPUs are a straightforward die shrink of the G80 to a 65nm process and were used in the GeForce 8800 GTS 512 (16 MPs, not to be confused with the "8800 GTS 384", a G80) and 9800-series cards, e.g., the 9800 GTX (16 MPs) and 9800 GX2 (two 9800 GTX's on a PCI Express bridge).

GPUs codenamed G84/G85/G86 are NVIDIA's low-end parts of the G80 series, with the same architecture but only 1–4 MPs and much lower memory

throughput. Similarly, G94/G96 describe low-end versions of the G92. These are never cost-effective for our purposes (except maybe testing code on the road).

Note: Manufacturers often sell a top-end, envelope-pushing chip at a huge markup, and slightly weaker chips (often just a batch failing a quality control binning) at far more reasonable prices. The lower-priced G80s (e.g., the "8800 GTS 384", 12 MPs, used in [46], or the older 8800 GT with 14 MPs) with slightly lower clock rates, fewer functional units, and lower memory throughput can achieve better price-performance ratio than the top-end G80s.

GT2xx Series. The GT2xx series started out at the same 65nm process as the G92, with a new and improved design. The GTX 260 and the GTX 280 both run at a slightly lower clock rate than the G92 but the GTX 260 has 24 MPs and the GTX 280 has 30 MPs, almost twice as many as the G92. GT2xx GPUs are also better in other ways. In particular, the size of the register file is doubled, which is very helpful for our implementation.

The GTX 285 and 295 were introduced early in 2009, shortly before the time of this writing. Our initial tests are consistent with reports that (a) the 285 is a simple die-shrink of the 280 to a 55nm process, and (b) the 295 is just two underclocked 285's bolted together.

3.2 The CUDA Programming Paradigm

CUDA provides an environment in which software programmers can program GPUs using a high-level, C-like language. A CUDA program (called a "kernel") starts with a source file `foo.cu`, which is first compiled by `nvcc`, the CUDA compiler, into code for a virtual machine (`foo.ptx`), then converted into actual machine code by the CUDA driver, and finally loaded and run on the GPU.

CUDA adopts a super-threaded, massively parallel computation model, in which computation is divided into many (typically thousands of) threads. A pool of physical processing units (e.g., the 128 SPs in G80) then executes these threads in a seemingly concurrent fashion. This time-sharing of physical units by many threads or computations is necessary because the instruction latency is high: a typical instruction takes 20–24 clock cycles to execute in its entirety. Because the SPs are fully pipelined, with enough instructions "in flight", we can hide this latency and approach the theoretical limit of one dispatched instruction per cycle per SP. CUDA manuals suggest a minimum of 192 threads per MP. This can be understood as 8 SPs $\times$ 24 stages $=$ 192 in order to hide instruction latency completely.

Modern CPUs do a lot more than pipelining. They actively search for independent instructions to issue in a program stream, *dispatching them out of order if needed*. However, out-of-order execution requires a lot of extra circuitry. NVIDIA has opted instead to make its chips completely in-order, hence CUDA mostly utilizes what is called *thread-level* (in contrast with instruction-level) parallelism.

At the programming level, the minimal scheduling entity is a *warp* of threads, which consists of 32 threads in the current version of CUDA. A warp must be

executed by a single MP. It takes four cycles for an MP to issue an instruction for a warp of threads (16 if the instruction is to be executed by the super function units). To achieve optimal instruction throughput, the threads belonging to the same warp must execute the same instruction, for there is only one instruction-decoding unit on each MP. We may hence regard an MP as a 32-way SIMD vector processor.

We note that the GPU threads are lightweight hardware threads, which incur little overhead in context switch. In order to support fast context switch, the physical registers are divided among all active threads. This creates pressure when programming GPUs. For example, on G80 and G92 there are only 8192 registers per MP. If we were to use 256 threads, then each thread could only use 32 registers, a tight budget for implementing complicated algorithms. The situation improved with the GT2xx family having twice as many registers, relieving the register pressure and making programming much easier.

To summarize, the massive parallelism in NVIDIA's GPU architecture makes programming on graphics cards very different from sequential programming on a traditional CPU. In general, GPUs are most suitable for executing the data-parallel part of an algorithm. Finally, to get the most out of the theoretical arithmetic throughput, one must minimize the number of memory accesses and meticulously arrange the parallel execution of hardware threads to avoid resource contention such as bank conflict in memory access.

3.3 Limitations and Alternatives

Race Conditions and Synchronization. A pitfall frequently encountered when programming multiple threads is race conditions. In CUDA, threads are organized into "blocks" so that threads belonging to the same block execute on the same MP and time-share the SPs on a per-instruction, round-robin fashion. Sometimes, the execution of a block of threads will need to be serialized when there is resource contention, e.g., when accessing device memory, or accessing shared memory when there is a bank conflict. Synchronization among a block of threads is achieved by calling the intrinsic `__syncthreads()` primitive, which blocks the execution until all threads in a block have reached the same point in the program stream. Another use of this primitive is to set up synchronization barriers. Without such barriers, the optimizing compiler can sometimes reorder the instructions too aggressively, resulting in race conditions when the code is executed concurrently by a block of threads.

Pressure on Fast On-die Memories. A critically limited GPU resource is memory — in particular, fast memory — including per-thread registers and per-MP shared memory. For example, on a G8x/G9x/G2xx GPU the per-SP working set of 2 kB is barely enough room to hold the base point and intermediate point for a scalar multiplication on an elliptic curve without any precomputation. To put this in perspective, all 240 SPs on a gigantic (1.4×10^9 gates) GTX 280 have between them 480 kB fast memory. That is less than the 512 kB of L2 cache in an aged Athlon 64 (1.6×10^8 gates)! Unfortunately, CUDA requires

many more (NVIDIA recommends 24 times the number of SPs) threads to hide instruction latency effectively. Therefore, we will need collaboration and hence communication among groups of threads in order to achieve a high utilization of the instruction pipelines when implementing modular arithmetic operations on GPUs.

A Brief Comparison to AMD GPUs. The main competition to NVIDIA's GeForce is Radeon from AMD (formerly ATI). The AMD counterpart to CUDA is Brook+, also a C/C++-derived language. Brook+ programming is similar to CUDA programming: GPU programs ("shaders") are compiled into intermediate code, which is converted on the fly into machine instructions. See Table 1 for a comparison between current GeForce and Radeon GPUs.

Table 1. Comparison of Leading NVIDIA and AMD Video Cards

NVIDIA/AMD Lead GPU Series	NVIDIA GT200	AMD RV770
Top configuration	GeForce GTX 295	Radeon 4870x2
Arithmetic clock	1250 MHz	750 MHz
Registers per MP (or SIMD core)	16k × 32-bit	16k × 128-bit
#MPs / #SPs	$2 \times (30 \times 8)$	$2 \times (10 \times 16(\times 5))$
Registers on each chip	491,520 (1.875MB)	163,840 (2.5MB)
Local store per MP/SIMD core	16 kB	16 kB
Global store per chip	None	16 kB
Max threads on chip	30,720	16,384
Max threads per MP	1,024	$> 1,000$

GeForce and Radeon have different hardware and software models. Recall that each GeForce MP has 8 SPs, each with a single-precision floating-point fused multiplier-adder, plus 2 super function units, which can dispatch 4 single-precision multiplications per cycle. The Radeon equivalent of an MP, called an "SIMD core", has 16 VLIW (very long instruction word) SPs, each of which is capable of delivering 5 single-precision floating-point operations every cycle. Pipelines on the Radeon are around a dozen stages deep, half as long as those on the GeForce.

Overall the two architectures pose similar challenges: there are many threads but very little fast memory available to each thread. A naïve calculation suggests that to hide the latency of arithmetic operations one must schedule $16 \times 12 = 192$ threads per SIMD core with a Radeon, and 192 threads per MP with a GeForce, so the number of registers per thread is similar for both architectures.

As a Radeon SIMD core does 80 floating-point operations per cycle to a GeForce MP's 20, but has at most 32 kB of scratch memory vs. 16 kB for the GeForce MP, one can expect that a program for a Radeon would be more storage-starved than for a GeForce. We plan to investigate Radeon cards as an alternative to CUDA and GeForce cards, but our initial estimate is that

ECM's storage pressure makes GeForce cards more suitable than Radeon cards for ECM.

4 High-Throughput Modular Arithmetic on a GPU

Modular arithmetic is the main bottleneck in computing scalar multiplication in ECM. In this section we describe our implementation of modular arithmetic on a GPU, focusing specifically on modular multiplication, the rate-determining mechanism in ECM. We will explain the design choices we have made and show how parallelism is used on this level.

4.1 Design Choices of Modular Multiplication

For our target of 280-bit integers, schoolbook multiplication needs less intermediate storage space and synchronization among cooperative threads than the more advanced algorithms such as Karatsuba. Moreover, despite requiring a smaller number of word multiplications, Karatsuba multiplication is slower on GPUs because there are fewer pairs of multiplications and additions that can be merged into single MAD instructions, resulting in a higher instruction count. It is partly for this reason that we choose to implement the modular multiplier using floating-point arithmetic as opposed to 32-bit integer arithmetic, which does not have the fused multiply-and-add instruction; another reason is that floating-point multiplication currently has a higher throughput on NVIDIA GPU than its 32-bit integer counterpart.

We represent an integer using L limbs in radix 2^r, with each limb stored as a floating-point number between -2^{r-1} and 2^{r-1}. This allows us to represent any integer between $-R/2$ and $R/2$, where $R = 2^{Lr}$. We choose to use Montgomery representation [37] of the integers modulo m, where m is the integer to be factored by ECM, and thus represent $x \bmod m$ as $x' \equiv Rx \pmod m$. Note that our limbs can be negative, so we use a signed representative in $-m/2 \leq (x' \bmod m) < m/2$. In Montgomery representation, addition and subtraction are performed on the representatives as usual. Let m' be the unique positive integer between 0 and R such that $RR' - mm' = 1$. Given $x' \equiv Rx \pmod m$ and $y' \equiv Ry \pmod m$ the multiplication is computed on the representatives as $\alpha = (x'y' \bmod R)m' \bmod R$ followed by $\beta = (x'y' + \alpha m)/R$. Note that since R is a power of 2, modular reductions modulo R correspond to taking the lower bits while divisions by R correspond to taking the higher bits. One verifies that $-m < \beta < m$ and $\beta \equiv R(xy) \pmod m$.

The Chosen Parameters. In the implementation described in this paper we take $L = 28$ and $r = 10$. Thus, we can handle integers up to around 280 bits. To fill up each MP with enough threads to effectively hide the instruction latency, we choose a block size of 256 threads; together such a block of threads is in charge of computing eight 280-bit arithmetic operations at a time. This means that we have an 8-way modular multiplier per MP. Each modular multiplication

needs three 280-bit integer multiplications: one to obtain $x'y'$, one to obtain α, and the third to obtain β. Each of the three integer multiplications is carried out by 28 threads, each of which is responsible for cross-multiplying 7 limbs from one operand with 4 from the other. The reason why we do not use all 32 threads is clear now: because $\gcd(7, 4) = 1$, there can never be any bank conflict or race condition in the final stage when these 28 threads are accumulating the partial products in shared memory. Bank conflicts, on the other hand, can still occur when threads are loading the limbs into their private registers before computing the partial products, so we carefully arrange x' and y' from different curves in shared memory, inserting appropriate padding when necessary, to avoid all bank conflicts in accessing shared memory.

4.2 Instruction Count Analysis

We give an estimate of the instruction count of our design on GPUs. Recall that, in our design, each thread is responsible for cross-multiplying the limbs in a 7-by-4 region. In the inner loop of integer multiplication, each thread needs to load these limbs into registers (11 loads from on-die shared memory), multiply and accumulate them into temporary storage (28 MAD instructions), and then accumulate the result in a region shared by all 28 threads. That last part includes 10 load-and-adds, 10 stores, and 10 synchronization barriers (`__syncthreads`) to prevent the compiler from reordering instructions incorrectly. Together, it should take 69 instructions per thread (plus other overhead) to complete such a vanilla multiplication. A partial parallel carry takes about 7 instructions by properly manipulating floating-point arithmetic instructions, and we need two partial carries in order to bring the value in each limb to its normal range. Furthermore, in Montgomery reduction we need a full carry for an intermediate result that is of twice the length, so we essentially need 4 full carries in each modular multiplication, resulting in 56 extra instructions per thread. This gives a total of 263 instructions per modular multiplication.

5 Fast ECM on a GPU

We now describe our implementation of ECM on a GPU using the modular multiplier described in the previous section. Recall that the speed bottleneck of ECM is scalar multiplication on an elliptic curve modulo m and that the factorization of m involves this computation on many curves.

Applications such as the NFS add a further dimension in that factorizations of many auxiliary numbers are needed. We decided to use the parallelism of the GPU to handle several curves for a given auxiliary integer, which can thus be stored in the shared memory of an MP. All SPs in an MP follow the same series of instructions which is a scalar multiplication on the respective curve modulo the same m and with the same scalar s. Different auxiliary factorizations inside NFS can be handled by different MPs in a GPU or different GPUs in parallel since no communication is necessary among the factorizations. For the rest of this section we consider one fixed m and s for the computation on a single MP.

Step	MAU 1	MAU 2	
1	$A{=}X_1^2$	$B{=}Y_1^2$	S
2	$X_1{=}X_1+Y_1$	$C{=}A+B$	a
3	$X_1{=}X_1^2$	$Z_1{=}Z_1^2$	S
4	$X_1{=}X_1-C$	$Z_1{=}Z_1+Z_1$	a
5	$B{=}B-A$	$Z_1{=}Z_1-C$	a
6	$X_1{=}X_1\times Z_1$	$Y_1{=}B\times C$	M
7	$A{=}X_1\times X_1$	$Z_1{=}Z_1\times C$	M
8	$Z_1{=}Z_1^2$	$B{=}Y_1^2$	S
9	$Z_1{=}Z_1+Z_1$	$C{=}A+B$	a
10	$B{=}B-A$	$X_1{=}X_1+Y_1$	a
11	$Y_1{=}B\times C$	$X_1{=}X_1\times X_1$	M
12	$B{=}Z_1-C$	$X_1{=}X_1-C$	a
13	$Z_1{=}B\times C$	$X_1{=}X_1\times B$	M
			4M+3S+6a

Fig. 1. Explicit formulas for DBL-DBL

The CPU first prepares the curve parameters (including the coordinates of the starting point) in an appropriate format and passes them to the GPU for scalar multiplication, whose result will be returned by the GPU. The CPU then does the gcd computation to determine whether we have found any factors.

Our implementation of modular arithmetic in essence turns an MP in a GPU into an 8-way modular arithmetic unit (MAU) that is capable of carrying out 8 modular arithmetic operations simultaneously. How to map our elliptic-curve computation onto this array of 8-way MAUs on a GPU is of crucial importance. We have explored two different approaches to use the 8-way MAUs we have implemented. The first one is straightforward: we compute on 8 curves in parallel, each of which uses a dedicated MAU. This approach results in 2 kB of working memory per curve, barely enough to store the curve parameters (including the base point) and the coordinates of the intermediate point. Besides the base point, we cannot cache any other points, which implies that the scalar multiplication can use only a non-adjacent form (NAF) representation of s. So we need to compute $\log_2 s$ doublings and on average $(\log_2 s)/3$ additions to compute $[s]\tilde{P}$.

In the second approach, we combine 2 MAUs to compute the scalar multiplication on a single curve. As mentioned in Sections 2 and 4, our implementation uses Montgomery representation of integers, so it does not benefit from multiplications with small values. In particular, multiplications with the curve coefficient d take the same time as general multiplications. We provide the base point and all precomputed points (if any) in affine coordinates, so all curve additions are mixed additions. Inspecting the explicit formulas, one notices that both addition and doubling require an odd number of multiplications/squarings. In order to avoid idle multiplication cycles, we have developed new parallel formulas that pipeline two group operations. The scalar multiplication can be composed of the building blocks DBL-DBL (doubling followed by doubling), mADD-DBL (mixed addition followed by doubling) and DBL-mADD. Note that there are never two

Step	MAU 1	MAU 2	
1	$B{=}x_2 \times Z_1$	$C{=}y_2 \times Z_1$	M
2	$A{=}X_1 \times Y_1$	$Z_1{=}B \times C$	M
3	$E{=}X_1 - B$	$F{=}Y_1 + C$	a
4	$X_1{=}X_1 + C$	$Y_1{=}Y_1 + B$	a
5	$E{=}E \times F$	$Y_1{=}X_1 \times Y_1$	M
6	$F{=}A + Z_1$	$B{=}A - Z_1$	a
7	$E{=}E - B$	$Y_1{=}Y_1 - F$	a
8	$Z_1{=}E \times Y_1$	$X_1{=}E \times F$	M
9	$Y_1{=}Y_1 \times B$	$A{=}X_1 \times X_1$	M
10	$Z_1{=}Z_1^2$	$B{=}Y_1^2$	S
11	$Z_1{=}Z_1 + Z_1$	$C{=}A + B$	a
12	$B{=}B - A$	$X_1{=}X_1 + Y_1$	a
13	$Y_1{=}B \times C$	$X_1{=}X_1 \times X_1$	M
14	$B{=}Z_1 - C$	$X_1{=}X_1 - C$	a
15	$Z_1{=}B \times C$	$X_1{=}X_1 \times B$	M
			7M+1S+7a

Fig. 2. Explicit formulas for mADD-DBL

subsequent additions. At the very end of the scalar multiplication, one might encounter a single DBL or mADD, in that case one MAU is idle in the final multiplication.

The detailed formulas are given in Fig. 1, Fig. 2, and Fig. 3. The input to all algorithms is the intermediate point, given in projective coordinates $(X_1 : Y_1 : Z_1)$; the algorithms involving additions also take a second point in affine coordinates (x_2, y_2) as input. The variables x_2, y_2 are read-only; the variables X_1, Y_1, Z_1 are modified to store the result. We have tested the formulas against those in the EFD [9] and ensured that there would be no concurrent reads/writes by testing the stated version and the one with the roles of MAU 1 and MAU 2 swapped. The horizontal lines indicate the beginning of the second operation. There are no idle multiplication stages and only in DBL-mADD there is a wait stage for an addition; another addition stage is used for a copy, which can be implemented as an addition $Z_1 = X_1 + 0$. So the pipelined algorithms achieve essentially perfect parallelism.

We note that in our current implementation, concurrent execution of a squaring and a multiplication does not result in any performance penalty since squaring is implemented as multiplication of the number by itself. Even if squarings could be executed somewhat faster than general multiplications the performance loss is minimal, e.g., instead of needing 3M+4S per doubling, the pipelined DBL-DBL formulas need 4M+3S per doubling.

We also kept the number of extra variables to a minimum. The pipelined versions need one extra variable compared to the versions on a single MAU but now two MAUs share the computation. This frees up enough memory so that we can store the eight points $\tilde{P}, [3]\tilde{P}, [5]\tilde{P}, \ldots, [15]\tilde{P}$ per curve. We store these points in affine coordinates using only two $\mathbf{Z}/m$ elements' worth of storage

Step	MAU 1	MAU 2	
1	$A{=}X_1^2$	$B{=}Y_1^2$	S
2	$X_1{=}X_1 + Y_1$	$C{=}A + B$	a
3	$X_1{=}X_1^2$	$Z_1{=}Z_1^2$	S
4	$X_1{=}X_1 - C$	$Z_1{=}Z_1 + Z_1$	a
5	$B{=}B - A$	$Z_1{=}Z_1 - C$	a
6	$X_1{=}X_1 \times Z_1$	$Y_1{=}B \times C$	M
7	$Z_1{=}Z_1 \times C$	$A{=}X_1 \times Y_1$	M
8	$B{=}x_2 \times Z_1$	$C{=}y_2 \times Z_1$	M
9	$E{=}X_1 - B$	$F{=}Y_1 + C$	a
10	$X_1{=}X_1 + C$	$Y_1{=}Y_1 + B$	a
11	$E{=}E \times F$	$Z_1{=}B \times C$	M
12	$F{=}A + Z_1$	$B{=}A - Z_1$	a
13	$E{=}E - B$	$Z_1{=}X_1$	a
14	$A{=}Z_1 \times Y_1$	$X_1{=}E \times F$	M
15	$A{=}A - F$		a
16	$Z_1{=}E \times A$	$Y_1{=}A \times B$	M
			6M+2S+8a

Fig. 3. Explicit formulas for DBL-mADD

space. With these precomputations we can use a signed-sliding-window method to compute $[s]\tilde{P}$. This reduces the number of mixed additions to an average of $(\log_2 s)/6$ (and worst case of $(\log_2 s)/5$).

6 Experimental Results

We summarize our results in Tables 2 and 3. Our experiments consist of running stage-1 ECM on the product of two 140-bit prime numbers with B_1 ranging from 2^{10} to 2^{20} on various CPUs and GPUs. For CPU experiments, we run GMP-ECM, the state-of-the-art implementation of ECM, whereas for GPU experiments, we run our GPU ECM implementation as described in Sections 4 and 5.

The first column of each table lists the coprocessors. The next three columns list their specifications: number of cores, clock frequency, and theoretical maximal arithmetic throughput (Rmax). Note that the Rmax figures tend to underestimate CPUs' computational power while overestimating GPUs' because CPUs have wider data paths and are better at exploiting instruction-level parallelism. Also, in calculating GPUs' Rmax, we exclude the contribution from texture processing units because we do not use them. The next two columns give the actual performance numbers derived from our measurements.

Table 2 includes an extra row, the first row, that does not correspond to any experiments we have performed. This row is extrapolated from the result of Szerwinski and Güneysu published in CHES 2008 [46]. In their result, the scalar in the scalar multiplications is 224 bits long, whereas in our experiments, it is 11797 bits long. Therefore, we have scaled their throughput by 224/11797 to fit

Table 2. Performance results of stage-1 ECM

Coprocessor	#Cores	Freq (GHz)	Rmax (GFLOPS)	Mulmods (10^6/sec)	Curves (1/sec)
CHES 2008 [46] (scaled)	96	1.2	230.4		26.81
8800 GTS (G80)	96	1.2	230.4	7.51	57.30
8800 GTS (G92)	128	1.625	416.0	13.64	104.14
GTX 260	192	1.242	476.9	14.97	119.05
GTX 280	240	1.296	622.1	19.53	155.29
Core 2 Duo E6850	2	3.0	48.0	7.85	75.17
Core 2 Quad Q6600	4	2.4	76.8	13.03	124.71
Core 2 Quad Q9550	4	2.83	90.7	14.85	142.17
GTX 260 (parallel)	192	1.242	476.9	16.61	165.58
GTX 280 (parallel)	240	1.296	622.1	22.66	216.78
GTX 295 (parallel)	480	1.242	1192.3	41.88	400.70
Q6600+GTX 295×2				96.79	926.11

into our context. We also note that their modulus is a special prime, which should lead to faster modular reduction, and that it only has 224 bits, as opposed to 280 in our implementation. We did not account for this difference in the performance figure stated. In spite of that, our implementation on the same platform achieves a significantly higher throughput, more than twice as many curves per second.

The remaining rows report two sets of performance numbers based on our cycle-accurate measurements of ECM execution time: per-second throughput of modular multiplication, and per-second throughput of elliptic-curve scalar multiplication with $B_1 = 8192$. For the GTX 260 and the GTX 280 we tried our ECM implementation using serial elliptic-curve arithmetic and our ECM implementation using parallel elliptic-curve arithmetic; both results are presented in the table. We are unable to make parallel arithmetic run on G80 and G92 because they do not have enough registers to accommodate the more complicated control code. The bottommost row represents the situation in which we use CPUs and GPUs simultaneously for ECM computations.

For the 8800 GTS (both G80 and G92), we used the CPU clock-cycle counter, so our scalar-multiplication measurements include the overhead of setting up the computation and returning the computed result. Our modular-multiplication measurements used separate experiments with $B_1 = 1048576$ to effectively eliminate this overhead. For the remaining GPUs we used the GPU clock-cycle counter, and used $B_1 = 8192$ in all cases to avoid overflow in the counter. By experimenting with additional choices of B_1 we have verified that, in all cases, modular-multiplication throughput roughly remains the same for different B_1's and thus can be used to accurately predict scalar-multiplication throughput given the number of modular multiplications executed in each scalar multiplication.

In Section 4.2 we have estimated that a modular multiplication needs at least 263 instructions. Take GTX 280 as an example: if we divide its Rmax in Table 2 by the achieved modular multiplication throughput, we see that in the experiment each

Table 3. Price-performance results of stage-1 ECM

Coprocessor	Component-wise Cost ($)	Component-wise performance/cost (1/(sec·$))	System-wise Cost ($)	System-wise performance/cost (1/(sec·$))
8800 GTS (G80)	119	0.48	1005	0.1140
8800 GTS (G92)	178	0.59	1123	0.1855
GTX 260	275	0.43	1317	0.1808
GTX 280	334	0.46	1435	0.2164
Core 2 Duo E6850	172	0.44	829	0.0907
Core 2 Quad Q6600	189	0.66	847	0.1472
Core 2 Quad Q9550	282	0.50	939	0.1541
GTX 260 (parallel)	275	0.60	1317	0.2515
GTX 280 (parallel)	334	0.65	1435	0.3021
GTX 295 (parallel)	510	0.79	2001	0.4005
Q6600+GTX 295×2	1210	0.77	2226	0.4160

modular multiplication consumes about 27454 floating-point operations, which can be delivered in 13727 GPU instructions. Given that 32 threads are dedicated to computing one single modular multiplication, each thread gets to execute about 429 instructions per modular multiplication. This number is about 60% more than what we have estimated. We believe that the difference is due to the fact that there are other minor operations such as modular additions and subtractions, as well as managerial operations like data movement and address calculations.

Table 3 shows price-performance figures for different ECM coprocessors. For each coprocessor, the next column shows the cheapest retail price pulled from on-line vendors such as NewEgg.com as of January 23, 2009, which in turn gives the per-US-dollar scalar-multiplication throughput listed in the next column. This price-performance ratio can be misleading because one could not compute ECM with a bare CPU or GPU — one needs a complete computer system with a motherboard, power supply, etc. In the last column we give the per-US-dollar scalar-multiplication throughput for an entire ECM computing system, based on the advice given by a web site for building computer systems of good price-performance ratio [6]. The baseline configuration consists of one dual-PCI-Express motherboard and one 750 GB hard drive in a desktop enclosure with a built-in 430-Watt power supply and several cooling fans. For CPU systems, we include the CPU, 8 GB of ECC RAM, and a low-price graphics card. In contrast, for GPU systems we include *two* identical graphics cards (since the motherboard can take two video cards). We also add a 750-Watt (1200-Watt in the case of GTX 295) power supply in order to provide enough power for the two graphics cards, plus a lower-priced Celeron CPU and 2 GB of ECC RAM. This is justified because when we use GPUs for ECM computation, we use the CPU only for light, managerial tasks. Finally, the configuration in the last row has both CPU and GPU working on ECM, which achieves the best price-performance ratio since the cost of the supporting hardware is shared by both CPU and GPUs.

We did not consider multi-socket motherboards with Opterons or Xeons because they are not competitive in price-performance ratio.

We conclude that although the Q6600 has a very good price-performance ratio among Intel CPUs — there is often such a "sweet spot" in market pricing for a high-end (but not quite highest-end) part, especially toward the end of the product life — the configuration of two GTX 295's achieves a superior price-performance ratio both component- and system-wise, not to mention that they can be aided by a CPU to achieve an even better price-performance ratio. The cost-effectiveness of GPUs for ECM makes GPUs suitable as a component of designs such as SHARK and allows ECM cofactorization to play a larger role inside the number-field sieve. To our knowledge, this is the first GPU implementation of elliptic-curve computation in which the GPU results are better than CPU results in the number of scalar multiplications per dollar and per second.

References

1. 13th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2005), Napa, CA, USA, April 17–20, 2005. IEEE Computer Society, Los Alamitos (2005); ISBN 0-7695-2445-1. See [44]
2. Aoki, K., Franke, J., Kleinjung, T., Lenstra, A.K., Osvik, D.A.: A Kilobit Special Number Field Sieve Factorization. In: ASIACRYPT 2007 [31], pp. 1–12 (2007) (Cited in §1, §1)
3. Atkin, A.O.L., Morain, F.: Finding suitable curves for the elliptic curve method of factorization. Mathematics of Computation 60, 399–405 (1993); ISSN 0025-5718, MR 93k:11115, `http://www.lix.polytechnique.fr/~morain/Articles/articles.english.html` (Cited in §2.2)
4. Bahr, F., Boehm, M., Franke, J., Kleinjung, T.: Subject: rsa200 (2005), `http://www.crypto-world.com/announcements/rsa200.txt` (Cited in §1)
5. Bahr, F., Franke, J., Kleinjung, T.: Discrete logarithms in GF(p) - 160 digits (2007), `http://www.nabble.com/Discrete-logarithms-in-GF(p)-----160-digits-td8810595.html` (Cited in §1)
6. Bernstein, D.J.: How to build the 2009.01.23 standard workstation, `http://cr.yp.to/hardware/build-20090123.html` (Cited in §6)
7. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards Curves. In: AFRICACRYPT [47], pp. 389–405 (2008), `http://eprint.iacr.org/2008/013` (Cited in §2.2)
8. Bernstein, D.J., Birkner, P., Lange, T., Peters, C.: ECM using Edwards curves (2008), `http://eprint.iacr.org/2008/016` (Cited in §2, §2.2, §2.2, §2.2)
9. Bernstein, D.J., Lange, T.: Explicit-formulas database (2008), `http://hyperelliptic.org/EFD` (Cited in §2.2, §5)
10. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: ASIACRYPT 2007 [31], pp. 29–50 (2007), `http://cr.yp.to/papers.html#newelliptic` (Cited in §2.2, §2.2)
11. Boneh, D. (ed.): CRYPTO 2003. LNCS, vol. 2729. Springer, Heidelberg (2003); ISBN 3-540-40674- 3. See [43]

12. Cavallar, S., Dodson, B., Lenstra, A.K., Leyland, P.C., Lioen, W.M., Montgomery, P.L., Murphy, B., te Riele, H., Zimmermann, P.: Factorization of RSA-140 Using the Number Field Sieve. In: ASIACRYPT 1999 [33], pp. 195–207 (1999) (Cited in §1)
13. Cavallar, S., Dodson, B., Lenstra, A.K., Lioen, W.M., Montgomery, P.L., Murphy, B., te Riele, H., Aardal, K., Gilchrist, J., Guillerm, G., Leyland, P.C., Marchand, J., Morain, F., Muffett, A., Putnam, C., Putnam, C., Zimmermann, P.: Factorization of a 512-Bit RSA Modulus. In: EUROCRYPT 2000 [41], pp. 1–18 (2000) (Cited in §1, §1)
14. Cook, D.L., Ioannidis, J., Keromytis, A.D., Luck, J.: CryptoGraphics: Secret Key Cryptography Using Graphics Cards. In: CT-RSA 2005 [36], pp. 334–350 (2005) (Cited in §3)
15. Cook, D.L., Keromytis, A.D.: CryptoGraphics: Exploiting Graphics Cards For Security. In: Advances in Information Security, vol. 20. Springer, Heidelberg (2006); ISBN 978-0- 387-29015-7 (Cited in §3)
16. Cowie, J., Dodson, B., Elkenbracht-Huizing, R.M., Lenstra, A.K., Montgomery, P.L., Zayer, J.: A World Wide Number Field Sieve Factoring Record: On to 512 Bits. In: ASIACRYPT 1996 [28], pp. 382–394 (1996) (Cited in §1)
17. Dwork, C. (ed.): CRYPTO 2006. LNCS, vol. 4117. Springer, Heidelberg (2006); ISBN 3-540- 37432-9. See [27]
18. Edwards, H.M.: A normal form for elliptic curves. Bulletin of the American Mathematical Society 44, 393–422 (2007), `http://www.ams.org/bull/2007-44-03/S0273-0979-07-01153-6/home.html` (Cited in §2.2)
19. Franke, J., Kleinjung, T., Paar, C., Pelzl, J., Priplata, C., Stahlke, C.: SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-Bit Integers. In: CHES 2005 [42], pp. 119–130 (2005) (Cited in §1, §1)
20. Gaj, K., Kwon, S., Baier, P., Kohlbrenner, P., Le, H., Khaleeluddin, M., Bachimanchi, R.: Implementing the Elliptic Curve Method of Factoring in Reconfigurable Hardware. In: CHES 2006 [23], pp. 119–133 (2006) (Cited in §1)
21. Galbraith, S.D. (ed.): Cryptography and Coding 2007. LNCS, vol. 4887. Springer, Heidelberg (2007); ISBN 978-3-540-77271-2. See [38]
22. Geiselmann, W., Shamir, A., Steinwandt, R., Tromer, E.: Scalable Hardware for Sparse Systems of Linear Equations, with Applications to Integer Factorization. In: CHES 2005 [42], pp. 131–146 (2005) (Cited in §1)
23. Goubin, L., Matsui, M. (eds.): CHES 2006. LNCS, vol. 4249. Springer, Heidelberg (2006); ISBN 3- 540-46559-6. See [20]
24. Hess, F., Pauli, S., Pohst, M.E. (eds.): ANTS 2006. LNCS, vol. 4076. Springer, Heidelberg (2006); ISBN 3- 540-36075-1. See [48]
25. Hisil, H., Wong, K., Carter, G., Dawson, E.: Faster group operations on elliptic curves (2007), `http://eprint.iacr.org/2007/441` (Cited in §2.2)
26. Joux, A., Lercier, R.: Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method, Mathematics of Computation 72, 953–967 (2003) (Cited in §1)
27. Joux, A., Lercier, R., Smart, N.P., Vercauteren, F.: The Number Field Sieve in the Medium Prime Case. In: CRYPTO 2006 [17], pp. 326–344 (2006) (Cited in §1)
28. Kim, K., Matsumoto, T. (eds.): ASIACRYPT 1996. LNCS, vol. 1163. Springer, Heidelberg (1996); ISBN 3-540-61872-4. See [16]
29. Kleinjung, T.: Cofactorisation strategies for the number field sieve and an estimate for the sieving step for factoring 1024-bit integers. In: Proceedings of SHARCS 2006 (2006), `http://www.math.uni-bonn.de/people/thor/cof.ps` (Cited in §1, §1)

30. Koblitz, N., Menezes, A.: Pairing-Based Cryptography at High Security Levels. In: Coding and Cryptography [45], pp. 13–36 (2005) (Cited in §1)
31. Kurosawa, K. (ed.): ASIACRYPT 2007. LNCS, vol. 4833. Springer, Heidelberg (2007); See [2], [10]
32. Laih, C.-S. (ed.): ASIACRYPT 2003. LNCS, vol. 2894. Springer, Heidelberg (2003); ISBN 3-540-20592-6. See [35]
33. Lam, K.-Y., Okamoto, E., Xing, C. (eds.): ASIACRYPT 1999. LNCS, vol. 1716. Springer, Heidelberg (1999); ISBN 3-540-66666-4. See [12]
34. Lenstra Jr., H.W.: Factoring integers with elliptic curves. Annals of Mathematics 126, 649–673 (1987); ISSN 0003-486X, MR 89g:11125, http://links.jstor.org/sici?sici=0003-486X(198711)2:126:3<649:FIWEC>2.0.CO;2-V (Cited §1)
35. Lenstra, A.K., Tromer, E., Shamir, A., Kortsmit, W., Dodson, B., Hughes, J., Leyland, P.C.: Factoring Estimates for a 1024-Bit RSA Modulus. In: ASIACRYPT 2003 [32], pp. 55–74 (2003) (Cited in §1)
36. Menezes, A.J. (ed.): CT-RSA 2005. LNCS, vol. 3376. Springer, Heidelberg (2005); ISBN 3- 540-24399-2. See [14]
37. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44, 519–521 (1985), http://www.jstor.org/pss/2007970 (Cited in §4.1)
38. Moss, A., Page, D., Smart, N.P.: Toward Acceleration of RSA Using 3D Graphics Hardware. In: Cryptography and Coding 2007 [21], pp. 364–383 (2007) (Cited in §3)
39. Oswald, E., Rohatgi, P. (eds.): CHES 2008. LNCS, vol. 5154. Springer, Heidelberg (2008); ISBN 978-3-540-85052-6. See [46]
40. Pelzl, J., Šimka, M., Kleinjung, T., Franke, J., Priplata, C., Stahlke, C., Drutarovský, M., Fischer, V., Paar, C.: Area-time efficient hardware architecture for factoring integers with the elliptic curve method. IEE Proceedings on Information Security 152, 67–78 (2005) (Cited in §1)
41. Preneel, B. (ed.): EUROCRYPT 2000. LNCS, vol. 1807. Springer, Heidelberg (2000); ISBN 3-540-67517-5. See [13]
42. Rao, J.R., Sunar, B. (eds.): CHES 2005. LNCS, vol. 3659. Springer, Heidelberg (2005); ISBN 3-540-28474-5. See [19], [22]
43. Shamir, A., Tromer, E.: Factoring Large Numbers with the TWIRL Device. In: CRYPTO 2003 [11], pp. 1–26 (2003) (Cited in §1)
44. Šimka, M., Pelzl, J., Kleinjung, T., Franke, J., Priplata, C., Stahlke, C., Drutarovský, M., Fischer, V.: Hardware Factorization Based on Elliptic Curve Method. In: FCCM 2005 [1], pp. 107–116 (2005) (Cited in §1)
45. Smart, N.P. (ed.): Cryptography and Coding 2005. LNCS, vol. 3796. Springer, Heidelberg (2005); See [30]
46. Szerwinski, R., Güneysu, T.: Exploiting the Power of GPUs for Asymmetric Cryptography. In: CHES 2008 [39], pp. 79–99 (2008) (Cited in §3.1, §6, §2)
47. Vaudenay, S. (ed.): AFRICACRYPT 2008. LNCS, vol. 5023. Springer, Heidelberg (2008); ISBN 978-3- 540-68159-5. See [7]
48. Zimmermann, P., Dodson, B.: 20 Years of ECM. In: ANTS 2006 [24], pp. 525–542 (2006) (Cited in §2)
49. Zimmermann, P.: 50 largest factors found by ECM, http://www.loria.fr/~zimmerma/records/top50.html (Cited in §1)

Double-Base Number System for Multi-scalar Multiplications

Christophe Doche[1,⋆], David R. Kohel[2], and Francesco Sica[3]

[1] Department of Computing, Macquarie University, Australia
doche@ics.mq.edu.au
[2] Université de la Mediterranée, Aix-Marseille II, France
kohel@maths.usyd.edu.au
[3] Department of Mathematics and Computer Science – AceCrypt
Mount Allison University, Sackville, Canada
fsica@mta.ca

Abstract. The Joint Sparse Form is currently the standard representation system to perform multi-scalar multiplications of the form $[n]P + m[Q]$. We introduce the concept of Joint Double-Base Chain, a generalization of the Double-Base Number System to represent simultaneously n and m. This concept is relevant because of the high redundancy of Double-Base systems, which ensures that we can find a chain of reasonable length that uses exactly the same terms to compute both n and m. Furthermore, we discuss an algorithm to produce such a Joint Double-Base Chain. Because of its simplicity, this algorithm is straightforward to implement, efficient, and also quite easy to analyze. Namely, in our main result we show that the average number of terms in the expansion is less than $0.3945 \log_2 n$. With respect to the Joint Sparse Form, this induces a reduction by more than 20% of the number of additions. As a consequence, the total number of multiplications required for a scalar multiplications is minimal for our method, across all the methods using two precomputations, $P + Q$ and $P - Q$. This is the case even with coordinate systems offering very cheap doublings, in contrast with recent results on scalar multiplications. Several variants are discussed, including methods using more precomputed points and a generalization relevant for Koblitz curves. Our second contribution is a new way to evaluate $\widehat{\phi}$, the dual endomorphism of the Frobenius. Namely, we propose formulae to compute $\pm\widehat{\phi}(P)$ with at most 2 multiplications and 2 squarings in the base field $\mathbb{F}_{2^d}$. This represents a speed-up of about 50% with respect to the fastest known techniques. This has very concrete consequences on scalar and multi-scalar multiplications on Koblitz curves.

Keywords: Elliptic curve cryptography, scalar multiplication, Double-Base Number System, Koblitz curves.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

⋆ This work was partially supported by ARC Discovery grant DP0881473.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 502–517, 2009.

1 Introduction

1.1 Elliptic Curve Cryptography

An *elliptic curve* defined over a field K can be seen as the set of points with coordinates in $\overline{K}$ lying on a cubic with coefficients in K. Additionally, the curve must be smooth, and if this is realized, the set of points on the curve can be endowed with an abelian group structure. This remarkable property has been exploited for about twenty years to implement fundamental public-key cryptographic primitives. We refer to [21] for a thorough, yet accessible, presentation of elliptic curves and to [2,16] for a discussion focused on cryptographic applications. In this context, two classes of elliptic curves are particularly relevant: those defined over a large prime field $\mathbb{F}_p$, represented either by a Weierstraß equation or an Edwards form [5] and Koblitz curves defined over $\mathbb{F}_2$.

The core operation in elliptic curve cryptography is a *scalar multiplication*, which consists in computing $[n]P$ given a point P on the curve and some integer n. Several methods exist relying on different representations of n. Among them, the *non-adjacent form* (NAF) allows to compute $[n]P$ with ℓ doublings and $\frac{\ell}{3}$ additions on average, where ℓ is the binary length of n.

1.2 Double-Base Number System

The *Double-Base Number System* (DBNS) was initially introduced by Dimitrov and Cooklev [10] and later used in the context of elliptic curve cryptography [11]. With this system, an integer n is represented as

$$n = \sum_{i=1}^{\ell} c_i 2^{a_i} 3^{b_i}, \text{ with } c_i \in \{-1, 1\}.$$

To find an expansion representing n, we can use a greedy-type algorithm whose principle is to find at each step the best approximation of a certain integer (n initially) in terms of a $\{2,3\}$-*integer*, *i.e.* an integer of the form $2^a 3^b$. Then compute the difference and reapply the process.

Example 1. *Applying this approach for* $n = 542788$*, we find that*

$$542788 = 2^8 3^7 - 2^3 3^7 + 2^4 3^3 - 2.3^2 - 2.$$

In [13], Dimitrov *et al.* show that for any integer n, this greedy approach returns a DBNS expansion of n having at most $O\big(\frac{\log n}{\log \log n}\big)$ terms. However, in general this system is not well suited for scalar multiplications. For instance, in order to compute $[542788]P$ from the DBNS expansion given in Example 1, it seems we need more than 8 doublings and 7 triplings unless we can use extra storage to keep certain intermediate results. But, if we are lucky enough that the terms in the expansion can be ordered in such a way that their powers of 2 and 3 are both decreasing, then it becomes trivial to obtain $[n]P$.

This observation leads to the concept of *Double-Base Chain* (DBC), introduced in [11], where we explicitly look for expansions such that $a_\ell \geqslant a_{\ell-1} \geqslant \cdots \geqslant a_1$ and $b_\ell \geqslant b_{\ell-1} \geqslant \cdots \geqslant b_1$. This guarantees that exactly a_ℓ doublings, b_ℓ triplings, $\ell - 1$ additions, and at most two registers are sufficient to compute $[n]P$. It is easy to modify the greedy algorithm to return a DBC. A tree-based algorithm has also been recently developed with the same purpose [14].

Example 2. *A modified version of the greedy algorithm returns the following DBC*

$$542788 = 2^{14}3^3 + 2^{12}3^3 - 2^{10}3^2 - 2^{10} + 2^6 + 2^2.$$

A DBC expansion is always longer than a DBNS one, but computing a scalar multiplication with it is now straightforward. The most natural method is probably to proceed from right-to-left. With this approach, each term $2^{a_i}3^{b_i}$ is computed individually and all the terms are added together. This can be implemented using two registers.

The left-to-right method, which can be seen as a Horner-like scheme, needs only one register. Simply initialize it with $[2^{a_\ell - a_{\ell-1}}3^{b_\ell - b_{\ell-1}}]P$, then add $c_{\ell-1}P$ and apply $[2^{a_{\ell-1} - a_{\ell-2}}3^{b_{\ell-1} - b_{\ell-2}}]$ to the result. Repeating this eventually gives $[n]P$, as illustrated with the chain of Example 2

$$[542788]P = [2^2]\big([2^4]\big([2^4]\big([3^2]\big([2^23]([2^2]P + P) - P\big) - P\big) + P\big) + P\big).$$

1.3 Multi-scalar Multiplication

A signature verification mainly requires a computation of the form $[n]P + [m]Q$. Obviously, $[n]P$, $[m]Q$ can be computed separately and added together at the cost of 2ℓ doublings and $\frac{2\ell}{3}$ additions on average, using the NAF and assuming that n and m are both of length ℓ. More interestingly, $[n]P + [m]Q$ can also be obtained as the result of a combined operation called a *multi-scalar multiplication*. So called Shamir's trick, a special case of an idea of Straus [20], allows to minimize the number of doublings and additions by jointly representing $\binom{n}{m}$ in binary. Scanning the bits from left-to-right, we perform a doubling at each step, followed by an addition of P, Q or $P + Q$ if the current bits of n and m are respectively $\binom{1}{0}$, $\binom{0}{1}$, or $\binom{1}{1}$. If $P + Q$ is precomputed, we see that $[n]P + [m]Q$ can be obtained with ℓ doublings and $\frac{3\ell}{4}$ additions, on average.

It is possible to do better, as shown by Solinas [19], using the redundancy and flexibility of signed-binary expansions. Indeed, the *Joint Sparse Form* (JSF) is a representation of the form

$$\binom{n}{m} = \begin{pmatrix} n_{\ell-1} & \dots & n_0 \\ m_{\ell-1} & \dots & m_0 \end{pmatrix}_{\text{JSF}}$$

such that the digits n_i, m_i fulfill certain conditions. Given two integers n and m, there is an efficient algorithm computing the JSF of n and m and if $\max(n, m)$ is of length ℓ, then the number of terms is at most $\ell + 1$ and the number of nonzero columns is $\frac{\ell}{2}$ on average. Also, the JSF is proven to be optimal, that is for any

given pair (n, m), the JSF has the smallest density among all joint signed-binary representations of n and m.

Example 3. *The joint sparse form of* $n = 542788$ *and* $m = 462444$ *is equal to*

$$\binom{n}{m} = \binom{100\bar{1}000100\bar{1}0100\bar{1}0\bar{1}00}{10000100100001000100}_{\text{JSF}}$$

where $\bar{1}$ *stands for* -1*. The computation of* $[n]P + [m]Q$ *requires 9 additions and 20 doublings, given that* $P + Q$ *and* $P - Q$ *are precomputed and stored.*

2 Joint Double-Base Number System

In the present article, we introduce the *Joint Double-Base Number System* (JDBNS) that allows to represent two integers n and m as

$$\binom{n}{m} = \sum_{i=1}^{\ell} \binom{c_i}{d_i} 2^{a_i} 3^{b_i}, \quad \text{with } c_i, d_i \in \{-1, 0, 1\}.$$

To compare with other representation systems, we define the *density* of a JDBNS expansion as the number of terms in the expansion divided by the binary length of $\max(n, m)$. It is easy to find an expansion with a very low density, however, just like in the one-dimension case, *cf.* Section 1.2, it cannot be used directly together with a Horner-like scheme. That is why we also introduce the concept of *Joint Double-Base Chain* (JDBC) where the sequences of exponents satisfy $a_\ell \geqslant a_{\ell-1} \geqslant \cdots \geqslant a_1$ and $b_\ell \geqslant b_{\ell-1} \geqslant \cdots \geqslant b_1$. With this additional constraint, the computation of $[n]P + [m]Q$ can be done very efficiently provided that the points $P + Q$ and $P - Q$ are precomputed.

Example 4. *A JDBC for* n *and* m *is as follows*

$$\begin{aligned}\binom{542788}{462444} = &\binom{1}{1}2^{14}3^3 + \binom{1}{0}2^{12}3^3 + \binom{\bar{1}}{1}2^9 3^3 + \binom{1}{1}2^9 3^2 + \binom{\bar{1}}{1}2^7 3^2 \\ &+ \binom{0}{1}2^6 3^2 + \binom{1}{\bar{1}}2^4 3^2 + \binom{1}{1}2^4 3 + \binom{0}{1}2^2 3 + \binom{1}{0}2^2.\end{aligned}$$

Based on this representation, it is now trivial to compute $[n]P + [m]Q$ *with a Horner-like scheme. Note however that the right-to-left method mentioned in Section 1.2 cannot be adapted in this context.*

Again, the greedy algorithm can be modified to return a JDBC, however, the resulting algorithm suffers from a certain lack of efficiency and is difficult to analyze. The method we discuss next is efficient, in the sense that it quickly produces very short chains, and simple allowing a detailed complexity analysis.

3 Joint Binary-Ternary Algorithm and Generalizations

In [8], Ciet *et al.* propose a *binary/ternary method* to perform a scalar multiplication by means of doublings, triplings, and additions. Let $v_p(x)$ denote the *p-adic valuation* of x, then the principle of this method is as follows. Starting from some integer n and a point P, divide n by $2^{v_2(n)}$ and perform $v_2(n)$ doublings, then divide the result by $3^{v_3(n)}$ and perform $v_3(n)$ triplings. At this point, we have some integer x that is coprime to 6. Thus setting $x = x - 1$ or $x = x + 1$ allows to repeat the process at the cost of an addition or a subtraction.

We propose to generalize this method in order to compute a JDBC. First, let us introduce some notation. For two integers x and y, we denote $\min\big(v_p(x), v_p(y)\big)$ by $v_p(x, y)$. It corresponds to the largest power of p that divides x and y *simultaneously*. Next, we denote by $\mathbb{X}$ the set of all pairs of positive integers (x, y) such that $v_2(x, y) = v_3(x, y) = 0$. Finally, for positive x and y, we introduce the function $\mathrm{gain}(x, y)$. We set $\mathrm{gain}(1, 1) = 0$, whereas for pairs $(x, y) \neq (1, 1)$, we define $\mathrm{gain}(x, y)$ as the largest factor $2^{v_2(x-c,y-d)}3^{v_3(x-c,y-d)}$ among all $c, d \in \{-1, 0, 1\}$.

For instance, $\mathrm{gain}(52, 45)$ is equal to 2^2, corresponding to $c = 0$ and $d = 1$. Note that this function can be implemented very efficiently, since in most cases it depends only on the remainders of x and y modulo 6.

3.1 Algorithm

Let us explain our generalization. Take two positive integers n and m. Divide by $2^{v_2(n,m)}3^{v_3(n,m)}$ in order to obtain $(x, y) \in \mathbb{X}$. The idea is then to call the function $\mathrm{gain}(x, y)$ and clear the common powers of 2 and 3 in $x - c$, $y - d$, where c and d are the coefficients maximizing this factor. (In case several pairs of coefficients achieve the same gain, any pair can be chosen.) The result is a new pair in $\mathbb{X}$ so that we can iterate the process, namely compute the corresponding

Algorithm 1. Joint Binary-Ternary representation

INPUT: Two integers n and m such that $n > 1$ or $m > 1$.
OUTPUT: A joint DB-Chain computing n and m simultaneously.

1. $i \leftarrow 1$ [current index]
2. $a_1 \leftarrow v_2(n, m)$ and $b_1 \leftarrow v_3(n, m)$ [common powers of 2 and 3]
3. $x \leftarrow n/(2^{a_1}3^{b_1})$ and $y \leftarrow m/(2^{a_1}3^{b_1})$ $[(x, y) \in \mathbb{X}]$
4. **while** $x > 1$ **or** $y > 1$ **do**
5. $\quad g \leftarrow \mathrm{gain}(x, y)$ [with coefficients c_i, d_i]
6. $\quad x \leftarrow (x - c_i)/g$ and $y \leftarrow (y - d_i)/g$ $[(x, y) \in \mathbb{X}]$
7. $\quad i \leftarrow i + 1$, $a_i \leftarrow a_i + v_2(g)$, and $b_i \leftarrow b_i + v_3(g)$
8. $c_i \leftarrow x$ and $d_i \leftarrow y$ $[c_i, d_i \in \{0, 1\}]$
9. **return** $\left(\binom{c_i}{d_i}2^{a_i}3^{b_i}\right)_{\mathrm{JDBC}}$

gain, divide by the common factor, and so on. Since x and y remain positive and decrease at each step, we will have at some point $x \leqslant 1$ and $y \leqslant 1$, causing the algorithm to terminate.

Example 5. *Algorithm 1 with input $n = 542788$ and $m = 462444$ returns the following expansion*

$$\binom{542788}{462444} = \binom{1}{1}2^{11}3^5 + \binom{1}{\bar{1}}2^9 3^4 + \binom{0}{1}2^7 3^4 + \binom{1}{\bar{1}}2^7 3^3 + \binom{0}{\bar{1}}2^5 3^3 + \binom{1}{1}2^5 3^2 - \binom{1}{1}2^5 3 + \binom{0}{1}2^4 + \binom{1}{\bar{1}}2^2.$$

Note that we have no control on the largest powers of 2 and 3 in the expansion returned by Algorithm 1. This can be a drawback since doublings and triplings have different costs in different coordinate systems. To have more control, simply modify the function gain. One possibility is to adjust the returned factor and coefficients depending on the remainders of x and y modulo a chosen constant. For instance, we can decide that when $x \equiv 4 \pmod{12}$ and $y \equiv 3 \pmod{12}$, then the gain should be 3 rather than 2^2. Doing that in each case, we can thus favor doublings or triplings. A complete analysis of the corresponding method is totally obvious. This is not the case for the general method that we address now.

3.2 Complexity Analysis

In the following, given integers n and m of a certain size, we compute the average density of a JDBC obtained with Algorithm 1, as well as the average values of the maximal powers of 2 and 3 in the joint expansion. This will in turn provide the average number of additions, doublings and triplings that are necessary to compute $[n]P + [m]Q$.

Let us start with the density of an expansion, which depends directly on the average number of bits cleared at each step of Algorithm 1. Given a pair $(x, y) \in \mathbb{X}$ and fixed values α and β, we determine the probability $p_{\alpha,\beta}$ that $\text{gain}(x, y) = 2^\alpha 3^\beta$ by enumerating the number of pairs having the desired gain in a certain square $\mathbb{S}$ and dividing by the total number of pairs in $\mathbb{X} \cap \mathbb{S}$. Let $\mathbb{S}_{\gamma,\delta}$ denote the square $[1, 2^\gamma 3^\delta]^2$. The total number of pairs we investigate is given by the following Lemma.

Lemma 1. *Given two integers γ and δ, the cardinality of $\mathbb{X} \cap \mathbb{S}_{\gamma,\delta}$, is equal to $2^{2\gamma+1}3^{2\delta-1}$.*

The proof is straightforward and is left to the reader. Next, let us choose γ, δ to actually compute $p_{\alpha,\beta}$. At first glance, it seems that the square $\mathbb{S}_{\alpha+1,\beta+1}$ is a good candidate for that. In fact, we can use it provided that when we consider a larger square, say $\mathbb{S}_{\alpha+\kappa+1,\beta+\eta+1}$, the number of pairs having a gain equal to $2^\alpha 3^\beta$ and the total number of pairs in $\mathbb{X}$ are both scaled by the same factor: $2^{2\kappa}3^{2\eta}$. Indeed, we expect that if (x, y) has a gain equal to $2^\alpha 3^\beta$, then all the pairs of

the form $(x+i2^{\alpha+1}3^{\beta+1}, y+j2^{\alpha+1}3^{\beta+1})$ with $(i,j) \in [0, 2^\kappa 3^\eta - 1]^2$ will have the same gain. However, this is not the case. For instance, $\text{gain}(26, 35) = 3^2$ whereas $\text{gain}(26 + 2 \times 3^3, 35 + 5 \times 2 \times 3^3) = 2^4$. These interferences are inevitable, but intuitively, they will become less and less frequent and will eventually disappear if we scan a set large enough. The following result makes this observation more precise.

Lemma 2. *Let α and β be two nonnegative integers. Take γ such that $2^\gamma > 2^\alpha 3^\beta$ and δ such that $3^\delta > 2^\alpha 3^\beta$. Then, for any $(x, y) \in \mathbb{X}$ whose gain is equal to $2^\alpha 3^\beta$, we have*

$$\text{gain}(x + i2^\gamma 3^\delta, y + j2^\gamma 3^\delta) = \text{gain}(x, y), \text{ for all } (i, j) \in \mathbb{Z}^2.$$

Lemma 2 gives a lower bound for $p_{\alpha,\beta}$. Indeed, let us consider a larger set $\mathbb{S}_{\gamma+\kappa,\delta+\eta}$. Then to any pair $(x, y) \in \mathbb{X} \cap \mathbb{S}_{\gamma,\delta}$ whose gain is $2^\alpha 3^\beta$, we can associate the elements $(x + i2^\gamma 3^\delta, y + j2^\gamma 3^\delta)$ with $(i, j) \in [0, 2^\kappa 3^\eta - 1]^2$ that are in $\mathbb{X} \cap \mathbb{S}_{\gamma+\kappa,\delta+\eta}$ and that have the same gain as (x, y). Conversely, if $(x_1, y_1) \in \mathbb{X} \cap \mathbb{S}_{\gamma+\kappa,\delta+\eta}$ and $\text{gain}(x_1, y_1) = 2^\alpha 3^\beta$, then (x_1, y_1) can be written $(x+i2^\gamma 3^\delta, y+j2^\gamma 3^\delta)$ with $(x, y) \in \mathbb{S}_{\gamma,\delta}$ and $\text{gain}(x, y) = \text{gain}(x_1, y_1)$. Overall, this ensures that scanning $\mathbb{S}_{\gamma,\delta}$ gives the exact probability for a pair to have a gain equal to $2^\alpha 3^\beta$ and allows to compute the first few probabilities. The following lemma deals with the remaining cases.

Lemma 3. *The probability $p_{\alpha,\beta}$ is bounded above by $\frac{1}{2^{2\alpha+1}3^{2\beta-3}}$ for any nonnegative α, β.*

The proofs of Lemmas 2 and 3 can be found in the extended version of the article available online [15]. We can now prove our main result.

Theorem 1. *Let $n \geqslant m$ be two integers such that $\gcd(n, m)$ is coprime with 6. The average density of the JDBC computing $\binom{n}{m}$ and returned by Algorithm 1 belongs to the interval $[0.3942, 0.3945]$. The average values of the biggest powers of 2 and 3 in the corresponding chain are approximately equal to $0.55 \log_2 n$ and $0.28 \log_2 n$.*

Proof. We determine the first probabilities $p_{\alpha,\beta}$ using Lemmas 1 and 2. Namely, we enumerate pairs having a gain equal to $2^\alpha 3^\beta$ in the square $\mathbb{S}_{\gamma,\delta}$, with γ and δ as in Lemma 2. With an appropriate implementation, we need to investigate only $2^{2(\gamma-\alpha)}3^{2(\delta+1-\beta)}$ pairs and a quick computation gives $p_{\alpha,\beta}$. We have performed the computations for $0 \leqslant \alpha \leqslant 8$ and $0 \leqslant \beta \leqslant 5$, and results show that these parameters cover more than 99.99% of the cases. We found that the probability $p_{i,j}$ is equal to $2^{-2i+3}3^{-2j}$ for $i \geqslant 2$ and $j \geqslant 1$. For $i \leqslant 1$ or $j = 0$, the probabilities do not seem to follow any pattern:

$$p_{0,0} = 0 \;\; p_{0,1} = \tfrac{2}{3^2} \;\; p_{0,2} = \tfrac{41}{2^3 3^4} \;\; p_{0,3} = \tfrac{169}{2^5 3^6} \;\; p_{0,4} = \tfrac{2729}{2^9 3^8} \;\; p_{0,5} = \tfrac{10921}{2^{11} 3^{10}}$$

$$p_{1,0} = 0 \; p_{1,1} = \tfrac{5}{3^3} \; p_{1,2} = \tfrac{95}{2^4 3^5} \; p_{1,3} = \tfrac{383}{2^6 3^7} \; p_{1,4} = \tfrac{6143}{2^{10} 3^9} \; p_{1,5} = \tfrac{24575}{2^{12} 3^{11}}$$

$$p_{2,0} = \tfrac{5}{2.3^2} \; p_{3,0} = \tfrac{7}{2^3 3^2} \; p_{4,0} = \tfrac{17}{2^3 3^4} \; p_{5,0} = \tfrac{635}{2^7 3^6} \; p_{6,0} = \tfrac{637}{2^6 3^6} \; p_{7,0} = \tfrac{2869}{2^{10} 3^8} \; p_{8,0} = \tfrac{51665}{2^{13} 3^{10}}.$$

Now, if the gain of (x, y) is equal to $2^{\alpha}3^{\beta}$ in Line 5 of Algorithm 1, then the sizes of x and y both decrease by $(\alpha + \beta \log_2 3)$ bits in Line 6. Therefore, if K denotes the average number of bits eliminated at each step of Algorithm 1, we have

$$K = \sum_{\alpha=0}^{\infty} \sum_{\beta=0}^{\infty} p_{\alpha,\beta}(\alpha + \beta \log_2 3) \geqslant \sum_{\alpha=0}^{8} \sum_{\beta=0}^{5} p_{\alpha,\beta}(\alpha + \beta \log_2 3)$$

and thanks to the values above, we obtain $K \geqslant 2.53519$. Using Lemma 3, we bound the remaining terms in the double sum to get $K \leqslant 2.53632$. The density being the inverse of K, we deduce the bounds of the theorem.

Similarly, we deduce that on average we divide by $2^{\overline{\alpha}}3^{\overline{\beta}}$ at each step for some $\overline{\alpha} \in [1.40735, 1.40810]$ and $\overline{\beta} \in [0.71158, 0.71183]$. To obtain the average of the largest power of 2 (respectively 3) in an expansion, we simply note that it is equal to $\overline{\alpha}$ (respectively $\overline{\beta}$) multiplied by the average length of the expansion.□

Since the JSF has a joint density of $\frac{1}{2}$, we see that a JDBC returned by Algorithm 1 has on average 21% less terms than a JSF expansion, whereas both representation systems require exactly 2 precomputations. See Table 2 to appreciate the impact of the Joint Binary-Ternary algorithm overall on multi-scalar multiplications.

3.3 Variants of the Joint Binary-Ternary Method

One simple generalization is to allow nontrivial coefficients in the expansion. This corresponds to use more precomputed points when computing a multi-scalar multiplication. For instance, if we allow the coefficients in the expansion to be $0, \pm 1, \pm 5$, then 10 points must be stored to compute $[n]P + [m]Q$ efficiently. Namely, $P+Q$, $P-Q$, $[5]P$, $[5]Q$, $[5]P+Q$, $[5]P-Q$, $P+[5]Q$, $P-[5]Q$, $[5]P+[5]Q$, and $[5]P-[5]Q$. The only difference with Algorithm 1 lies in the function gain(x, y), which now computes the largest factor $2^{v_2(x-c,y-d)}3^{v_3(x-c,y-d)}$ for $c, d \in \{-5, -1, 0, 1, 5\}$. Clearly, the average number of bits that is gained at each step is larger than in Algorithm 1, and indeed, following the ideas behind Theorem 1, it is possible to show that the average density of an expansion returned by this variant is approximately equal to 0.3120. Note that this approach gives shorter multi-chains on average than the hybrid method explained in [1] that uses 14 precomputations for a density of 0.3209.

If we want to add a new value, *e.g.* 7, to the set of coefficients, we have to use 22 precomputed points, which does not seem realistic. If the computations are performed on a device with limited memory, storing 10 points is already too much. A possibility is to precompute only $P+Q$, $P-Q$, $[5]P$, and $[5]Q$ and use only coefficients of the form $\binom{1}{0}$, $\binom{0}{1}$, $\binom{1}{1}$, $\binom{1}{\bar{1}}$, $\binom{5}{0}$, $\binom{0}{5}$ and their opposite in the JDBC. In this scenario, adding a new coefficient has a moderate impact on the total number of precomputations. Again, the only difference lies in the function gain. It is easy to perform an analysis of this method following the steps that lead to Theorem 1. This is left to the interested reader.

Another variant, we call the *Tree-Based Joint Binary-Ternary method*, is a generalization of the tree-based approach to compute single DB-Chains [14]. Namely, instead of selecting the coefficients c, d that give the maximal gain in order to derive the next pair of integers, simply build a tree containing nodes (x, y) corresponding to all the possible choices of coefficients. The idea is that taking a maximal gain at each step is not necessarily the best choice overall. Giving a certain flexibility can allow to find shorter expansions. The downside is that the number of nodes grows exponentially so that the algorithm becomes quickly out of control. A practical way to deal with this issue is to trim the tree at each step, by keeping only a fixed number B of nodes, for instance the B smallest ones (*e.g.* with respect to the Euclidean norm). Tests show that the value B does not have to be very large in order to introduce a significant gain. In practice, we use $B = 4$, which achieves a good balance between the computation time and the quality of the chain obtained.

Algorithm 2. Tree-Based Joint Binary-Ternary method

INPUT: Two integers n and m such that $n > 1$ or $m > 1$ and a bound B.
OUTPUT: A tree containing a joint DB-chain computing n and m.

1. Initialize a tree $\mathcal{T}$ with root node (n, m)
2. **if** $v_2(n, m) > 0$ or $v_3(n, m) > 0$ **then**
3. $\quad g \leftarrow 2^{v_2(n,m)}3^{v_3(n,m)}$
4. $\quad$ Insert the child $\left(\frac{n}{g}, \frac{m}{g}\right)$ under the node (n, m)
5. **repeat**
6. $\quad$ **for** each leaf node $\mathcal{L} = (x, y)$ in $\mathcal{T}$ **do** [insert 8 children]
7. $\quad\quad$ **for** each pair $(c, d) \in \{-1, 0, 1\}^2 \setminus \{(0, 0)\}$ **do**
8. $\quad\quad\quad g_{c,d} \leftarrow 2^{v_2(x-c,y-d)}3^{v_3(x-c,y-d)}$
9. $\quad\quad\quad \mathcal{L}_{c,d} \leftarrow \left(\frac{x-c}{g_{c,d}}, \frac{y-d}{g_{c,d}}\right)$ and insert $\mathcal{L}_{c,d}$ under $\mathcal{L}$
10. $\quad$ Discard any redundant leaf node
11. $\quad$ Discard all but the B smallest leaf nodes
12. **until** a leaf node is equal to $(1, 1)$
13. **return** $\mathcal{T}$

Remarks 6

(i) The choice $B = 1$ corresponds to the Joint Binary-Ternary method. It is clear that on average, the larger B is, the shorter will be the expansion. However, a precise complexity analysis of Algorithm 2 seems rather difficult.
(ii) To find an actual JDBC computing n and m, go through the intermediate nodes of any branch having a leaf node equal to $(1, 1)$.

(iii) To select the nodes that we keep in Line 11, we use a weight function that is in our case simply the size of the gain, of the form $2^\alpha 3^\beta$. To have more control on the largest powers of 2 and 3 in the expansion, we can use another weight function, *e.g.* depending on α or β.
(iv) It is straightforward to mix the tree-based approach with the use of nontrivial coefficients. Simply, modify Line 7 to handle different sets of coefficients.

Example 7. *To compute* $[542788]P+[462444]Q$*, the JSF needs 9 additions and 20 doublings, whereas the joint Binary-Ternary method only requires 8 additions, 11 doublings, and 5 triplings, cf. Examples 1 and 5. Applying Algorithm 2 with* $B = 4$*, we find that*

$$\binom{542788}{462444} = \binom{1}{1}2^{11}3^5 + \binom{1}{\bar{1}}2^9 3^4 + \binom{1}{1}2^6 3^4 + \binom{\bar{1}}{1}2^4 3^4 - \binom{1}{1}2^3 3^3 + \binom{\bar{1}}{0}2^2 3^2 + \binom{1}{\bar{1}}2^2 3 + \binom{1}{0}2^2.$$

This last expansion still requires 11 doublings and 5 triplings but saves one addition.

Next, we describe some experiments aimed at comparing all these methods in different situations.

3.4 Experiments

We have run some tests to compare the different methods discussed so far for different sizes ranging from 192 to 512 bits. More precisely, we have investigated the Joint Sparse Form (JSF), the Joint Binary-Ternary (JBT), and its Tree-Based variant with parameter B adjusted to 4 (Tree-JBT). All these methods require only 2 precomputations. Also, for the same set of integers, we have looked at methods relying on more precomputed values. The variant of the Tree-Based explained above that needs only $[5]P$ and $[5]Q$ on top of $P+Q$ and $P-Q$ is denoted Tree-JBT$_5$. In this spirit Tree-JBT$_7$ needs extra points $[7]P$ and $[7]Q$, whereas Tree-JBT$_{5^2}$ needs all the possible combinations, such as $[5]P+[5]Q$, that is 10 precomputations in total. Table 1 displays the different parameters for each method, in particular the length of the expansion, corresponding to the number of additions and the number of doublings and triplings. The values obtained are inline with those announced in Theorem 1. The notation $\#\mathcal{P}$ stands for the number of precomputed points required by each method.

To demonstrate the validity of our approach, we compare it against the Joint Sparse Form and the hybrid method [1]. These methods have the best known density when using respectively two and 14 precomputed points. Furthermore, we performed computations using *inverted Edwards coordinates* [7]. This system offers so efficient doublings that it makes a Double-Base approach irrelevant for single scalar multiplications [6]. Indeed, with this system a doubling can be

Table 1. Parameters of JDBC obtained by various methods

	Size	192 bits			256 bits			320 bits			384 bits			448 bits			512 bits		
Method	$\#\mathcal{P}$	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	$b_\ell\ell$	a_ℓ	b_ℓ	
JSF	2	96	192	0	128	256	0	160	320	0	190	384	0	224	448	0	256	512	0
JBT	2	77	104	55	102	138	74	128	174	92	153	208	110	179	241	130	204	279	146
Tree-JBT	2	72	107	53	96	141	72	119	178	89	143	214	107	167	248	126	190	281	145
Tree-JBT$_5$	4	64	105	54	85	141	71	106	176	90	126	211	108	147	246	126	169	281	145
Tree-JBT$_7$	6	60	102	55	80	137	74	99	171	93	119	204	112	139	238	131	158	273	150
Tree-JBT$_{52}$	10	54	105	54	72	140	72	89	176	90	107	210	109	125	245	127	142	283	144
Hybrid	14	61	83	69	82	110	92	102	138	115	123	165	138	143	193	161	164	220	184

obtained with 3M + 4S, a mixed addition with 8M + S, and a tripling with 9M + 4S, where M and S represent respectively a multiplication and a squaring in $\mathbb{F}_p$. To ease the comparisons, we make the usual assumption that 1S $\approx$ 0.8M.

Table 2 gives the overall number of multiplications needed for a scalar multiplication with a particular method, using inverted Edwards coordinates. These tests show that the Joint Binary-Ternary is faster than the Joint Sparse Form. The Tree-Based variant is even faster, but the time necessary to derive the expansion is considerably higher than the simple Joint Binary-Ternary. Beyond the speed-up, that is close to 5%, it is interesting to notice that even with very cheap doublings, Double-Base like methods are faster. Regarding methods requiring more precomputed values, it is to be noted that all the variants introduced in this paper use significantly less precomputed points than the hybrid method. Nevertheless, they are all faster when counting the costs of precomputations, as shown in Table 2. One can check that this is still the case even when those costs are ignored.

Table 2. Complexity of various scalar multiplication methods for different sizes

	Size	192 bits	256 bits	320 bits	384 bits	448 bits	512 bits
Method	$\#\mathcal{P}$	N_{M}	N_{M}	N_{M}	N_{M}	N_{M}	N_{M}
JSF	2	2044	2722	3401	4062	4758	5436
JBT	2	2004	2668	3331	3995	4664	5322
Tree-JBT	2	1953	2602	3248	3896	4545	5197
Tree-JBT$_5$	4	1920	2543	3168	3792	4414	5042
Tree-JBT$_7$	6	1907	2521	3137	3753	4365	4980
Tree-JBT$_{52}$	10	1890	2485	3079	3677	4270	4862
Hybrid	14	2047	2679	3311	3943	4575	5207

To conclude, note that JDBC expansions are relevant for scalar multiplications as well. Namely, if we want to compute $[n]P$, one possibility is to split n as $n_0 + 2^A 3^B n_1$, where A and B are fixed constants chosen so that n_0 and n_1 have approximately the same size and also to adjust the number of doublings and triplings. Then run Algorithm 1 or 2 to find a chain computing n_0 and n_1 simultaneously and derive $[n]P$ as $[n_0]P + [n_1]Q$, where $Q = [2^A 3^B]P$.

4 Koblitz Curves

The results above can be applied to compute a scalar multiplication on any elliptic curve. However, in practice, these techniques concern mainly curves defined over a prime field of large characteristic.

For Koblitz curves,

$$E_{a_2} : y^2 + xy = x^3 + a_2x^2 + 1, \quad a_2 \in \{0, 1\}$$

there exists a nontrivial endomorphism, the *Frobenius* denoted by ϕ and defined by $\phi(x, y) = (x^2, y^2)$. Let $\mu = (-1)^{1-a_2}$, then it is well-known that the Frobenius satisfies $\phi^2 - \mu\phi + [2] = [0]$. So, in some sense, the complex number τ such that $\tau^2 - \mu\tau + 2 = 0$ represents ϕ. If an integer n is equal to some polynomial in τ, then the endomorphism $[n]$ will be equal to the same polynomial in ϕ. The elements of the ring $\mathbb{Z}[\tau]$, called *Kleinian integers* [12], thus play a key role in scalar multiplications on Koblitz curves.

4.1 Representation of Kleinian Integers

It is easy to show that $\mathbb{Z}[\tau]$ is an Euclidean ring and thus any element $\eta \in \mathbb{Z}[\tau]$ has a τ*-adic representation* of the form

$$\eta = \sum_{i=0}^{\ell-1} c_i \tau^i, \quad \text{with } c_i \in \{0, 1\}.$$

There are also signed-digit representations and among them, the τ-NAF has a distinguished status, achieving an optimal density of $\frac{1}{3}\cdot$ Its generalization, the τ-NAF$_w$, has an average density of $\frac{1}{w+1}$ for $2^{w-2} - 1$ precomputed points.

In [3,4,12], the concept of Double-Base is extended to Kleinian integers. In particular, for a given $\eta \in \mathbb{Z}[\tau]$, there is an efficient algorithm described in [3] that returns a τ-DBNS expansion of the form

$$\eta = \sum_{i=1}^{\ell} \pm\tau^{a_i} z^{b_i},$$

where $z = 3$ or $\bar{\tau}$. This method produces in general an expansion whose terms cannot be ordered such that $a_\ell \geqslant a_{\ell-1} \geqslant \cdots \geqslant a_1$ and $b_\ell \geqslant b_{\ell-1} \geqslant \cdots \geqslant b_1$. Unlike what we have seen in Section 1.2, such an expansion can still be used to compute a scalar multiplication in certain situations. The price to pay is to incorporate conversion routines between polynomial and normal bases [18] to compute repeated applications of the Frobenius for free. This approach is described in [17].

Since implementing these conversion techniques can be challenging, especially on devices with limited capabilities, we will not follow this path and introduce instead the concept of τ*-Double-Base Chains* (τ-DBC) where, as in the integer

case, we ask that $a_\ell \geqslant a_{\ell-1} \geqslant \cdots \geqslant a_1$ and $b_\ell \geqslant b_{\ell-1} \geqslant \cdots \geqslant b_1$ in the expansion above. The algorithm described in [3] could be adapted to return a τ-DBC, however the implementation would certainly be tricky and the analysis quite involved. Instead, we can generalize the greedy algorithm or the binary-ternary method to produce such a chain.

4.2 Scalar Multiplication

The τ-adic representation of η implies that

$$[\eta]P = \sum_{i=0}^{\ell-1} c_i \phi^i(P).$$

Now, if we work in the extension $\mathbb{F}_{2^d}$, and if $\eta \in \mathbb{Z}$ is of size 2^d, the length ℓ of the τ-adic expansion of η is twice as long as what we expect, that is $2d$ instead of d. That is why in practice, we first compute $\delta = \eta \bmod \frac{\tau^d-1}{\tau-1}$. Under appropriate conditions, we have $[\delta]P = [\eta]P$ with δ of length half then length of η. From now on, we assume that this reduction has been done and that the length of η is approximately d.

Computing $[\eta]P$ with the τ-NAF involves $\frac{d}{3}$ additions on average and d Frobenius that need at most $3d$ squarings in $\mathbb{F}_{2^d}$. With the τ-NAF$_w$, we need $2^{w-2} - 1 + \frac{d}{w+1}$ additions, the same amount of Frobenius, and some memory to store $2^{w-2} - 1$ precomputed points. The complexity of the τ-DBNS is well understood, however as mentioned earlier, it requires change of basis techniques that are not available in our scenario.

The complexity of the τ-DBC is much more difficult to analyze. Only some experiments give an indication of its performance, and tests show that the τ-DBC cannot compete, for instance with the τ-NAF. The problem comes from the cost of the second endomorphism that is too expensive to balance the saving induced on the number of additions. To make use of the τ-DBC, it is crucial to reduce this cost. There is little hope to reduce significantly the cost of a tripling, that is why we focus our efforts on $\widehat{\phi}$.

Obviously, we can implement $\widehat{\phi}(P) = \mu P - \phi(P)$ with a subtraction and, in López–Dahab coordinates, this corresponds to the cost of a mixed addition, *i.e.* 8M + 5S, where M and S are respectively the cost of a multiplication and a squaring in $\mathbb{F}_{2^d}$. But it is possible to do better. Indeed, we can replace $\widehat{\phi}$ by the halving map using the equation $\phi\widehat{\phi}(P) = [2]P$. A halving works on the point $P = (x_1, y_1)$ represented as (x_1, λ_1) with $\lambda_1 = x_1 + y_1/x_1$. It involves solving a quadratic equation, computing a square root and a trace, and performing at least one multiplication, *cf.* [2]. It is thus difficult to accurately analyze the cost of a halving, but half the cost of a mixed López–Dahab addition, that is 4M + 4S, is a reasonable estimate. This is still too expensive to justify the use of the τ-DBC to compute a scalar multiplication. We show next how to compute $\pm\widehat{\phi}(P)$ in a much more efficient way.

4.3 Fast Evaluation of $\widehat{\phi}$

In this part, we show how to compute $\pm\widehat{\phi}(P)$ in López–Dahab coordinates with $2\mathrm{M}+\mathrm{S}$ when $a_2=1$ and $2\mathrm{M}+2\mathrm{S}$ when $a_2=0$.

Lemma 4. *Let $P_1=(X_1:Y_1:Z_1)$ be a point in López–Dahab coordinates on the curve E_{a_2} and let $P_2=\widehat{\phi}(P_1)$. Then the López–Dahab coordinates of P_2, namely $(X_2:Y_2:Z_2)$ satisfy*

$$X_2=(X_1+Z_1)^2,\quad Z_2=X_1Z_1,$$
$$Y_2=\big(Y_1+(1-a_2)X_2\big)\big(Y_1+a_2X_2+Z_2\big)+(1-a_2)Z_2^2.$$

The coordinates of the negative of P_2 are equal to $(X_2:Y_2':Z_2)$ with $Y_2'=\big(Y_1+a_2X_2\big)\big(Y_1+(1-a_2)X_2+Z_2\big)+(1-a_2)Z_2^2$.

The proof can be found in the extended version of the article [15].

This new way to compute $\widehat{\phi}$ is also beneficial to the τ-DBNS, especially regarding the algorithm described in [3]. A direct application of the formulae above induces a speed-up on the overall scalar multiplication ranging from 15% to 20%.

4.4 Multi-scalar Multiplication Algorithms

To perform $[\eta]P+[\kappa]Q$ at once, there is also a notion of τ-adic Joint Sparse Form, τ-JSF [9]. The τ-JSF and the JSF have very similar definitions. For instance, they have the same average joint density, that is $\frac{1}{2}\cdot$ However the optimality of the JSF does not carry over to the τ-JSF. Namely, for certain pairs in $\mathbb{Z}[\tau]$, the joint density of the τ-JSF expansion is not minimal across all the signed τ-adic expansions computing this pair.

Now, let us explain how we can produce joint τ-DBNS expansions and more importantly joint τ-DBC. The generalization of the greedy-type method is straightforward. At each step, find the closest approximation of (η,κ) of the form $(c\tau^\alpha\overline{\tau}^\beta, d\tau^\alpha\overline{\tau}^\beta)$ with $c,d\in\{-1,0,1\}$ with respect to the distance $d\big((\eta,\kappa),(\eta',\kappa')\big)=\sqrt{\mathrm{N}(\eta-\eta')^2+\mathrm{N}(\kappa-\kappa')^2}$, where $\mathrm{N}(\cdot)$ is the norm in $\mathbb{Z}[\tau]$. Then subtract the closest approximation and repeat the process until we reach $(0,0)$. To find a joint τ-DBC, do the same except that this search must be done under constraint, just like in the integer case.

Another possibility is to adapt the method developed in Section 3. We call this approach the Joint-$\tau\overline{\tau}$ method. The framework is exactly the same, the only difference lies in the function gain. This time gain(η,κ) computes a suitable common factor $\tau^\alpha\overline{\tau}^\beta$ of the elements $(\eta-c,\kappa-d)$ for $c,d\in\{-1,0,1\}$. We are not interested in the factor having the largest norm, instead we prefer to control the largest power of $\overline{\tau}$, as this has a crucial impact on the overall complexity. This can be done quite easily by adjusting certain parameters as it is suggested at the end of Section 3.1. For each choice of the function gain, there is a corresponding algorithm, that should be analyzed quite easily, following the integer case. We decided to run some experiments first to inform us on the optimal choices for the function gain. A summary is detailed next.

4.5 Experiments

We have run some tests to compare the τ-JSF with the Joint-$\tau\overline{\tau}$ for popular sizes used with Koblitz curves, ranging from 163 to 571 bits. Table 3 displays the different parameters for each method, in particular the length of the expansion, the values a_ℓ and b_ℓ corresponding respectively to the number of additions, the number of applications of ϕ and of $\widehat{\phi}$, as well as the total number of multiplications N_{M} in $\mathbb{F}_{2^d}$ needed to perform a multi-scalar multiplication for the corresponding size. Both methods require only 2 precomputations and the figures include those costs. Also to ease comparisons we have made the usual assumption that $1\mathrm{S} \approx 0.1\mathrm{M}$. Results show that our approach introduces improvements regarding scalar multiplications of 8 to 9% in total over the τ-JSF.

Table 3. Comparison between the τ-JSF and the Joint-$\tau\overline{\tau}$

	163 bits			233 bits			283 bits			347 bits			4409 bits			571 bits		
Method	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	$b_\ell\ell$	a_ℓ	b_ℓ	
τ-JSF	82	163	0	117	233	0	142	283	0	174	347	0	205	409	0	286	571	0
N_{M}		738			1050			1272			1558			1834			2555	
Joint-$\tau\overline{\tau}$	65	116	44	92	167	62	112	204	76	137	251	93	161	295	110	224	412	155
N_{M}		671			955			1154			1410			1665			2318	

References

1. Adikari, J., Dimitrov, V., Imbert, L.: Hybrid Binary-Ternary Joint Sparse Form and its Application in Elliptic Curve Cryptography, `http://eprint.iacr.org/2008/`
2. Avanzi, R.M., Cohen, H., Doche, C., Frey, G., Nguyen, K., Lange, T., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton (2005)
3. Avanzi, R.M., Dimitrov, V.S., Doche, C., Sica, F.: Extending scalar multiplication using double bases. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 130–144. Springer, Heidelberg (2006)
4. Avanzi, R.M., Sica, F.: Scalar multiplication on koblitz curves using double bases. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 131–146. Springer, Heidelberg (2006)
5. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
6. Bernstein, D.J., Birkner, P., Lange, T., Peters, C.: Optimizing double-base elliptic-curve single-scalar multiplication. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 167–182. Springer, Heidelberg (2007)
7. Bernstein, D.J., Lange, T.: Explicit-formulas database, `http://www.hyperelliptic.org/EFD/`
8. Ciet, M., Joye, M., Lauter, K., Montgomery, P.L.: Trading Inversions for Multiplications in Elliptic Curve Cryptography. Des. Codes Cryptogr. 39(2), 189–206 (2006)

9. Ciet, M., Lange, T., Sica, F., Quisquater, J.-J.: Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 388–400. Springer, Heidelberg (2003)
10. Dimitrov, V.S., Cooklev, T.: Hybrid Algorithm for the Computation of the Matrix Polynomial $I+A+\cdots+A^{N-1}$. IEEE Trans. on Circuits and Systems 42(7), 377–380 (1995)
11. Dimitrov, V.S., Imbert, L., Mishra, P.K.: Efficient and secure elliptic curve point multiplication using double-base chains. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 59–78. Springer, Heidelberg (2005)
12. Dimitrov, V.S., Järvinen, K.U., Jacobson Jr., M.J., Chan, W.F., Huang, Z.: FPGA implementation of point multiplication on koblitz curves using kleinian integers. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 445–459. Springer, Heidelberg (2006)
13. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: An Algorithm for Modular Exponentiation. Information Processing Letters 66(3), 155–159 (1998)
14. Doche, C., Habsieger, L.: A tree-based approach for computing double-base chains. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 433–446. Springer, Heidelberg (2008)
15. Doche, C., Kohel, D.R., Sica, F.: Double-Base Number System for Multi-Scalar Multiplications, `http://eprint.iacr.org/2008/`
16. Hankerson, D., Menezes, A.J., Vanstone, S.A.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2003)
17. Okeya, K., Takagi, T., Vuillaume, C.: Short memory scalar multiplication on koblitz curves. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 91–105. Springer, Heidelberg (2005)
18. Park, D.J., Sim, S.G., Lee, P.J.: Fast scalar multiplication method using change-of-basis matrix to prevent power analysis attacks on koblitz curves. In: Chae, K.-J., Yung, M. (eds.) WISA 2003. LNCS, vol. 2908, pp. 474–488. Springer, Heidelberg (2004)
19. Solinas, J.A.: Low-weight binary representations for pairs of integers. Combinatorics and Optimization Research Report CORR 2001-41, University of Waterloo (2001)
20. Straus, E.G.: Addition chains of vectors (problem 5125). Amer. Math. Monthly 70, 806–808 (1964)
21. Washington, L.C.: Elliptic Curves. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton (2003); number theory and cryptography

Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves

Steven D. Galbraith[1,⋆], Xibin Lin[2,⋆⋆], and Michael Scott[3,⋆⋆⋆]

[1] Mathematics Department,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX,
United Kingdom
steven.galbraith@rhul.ac.uk
[2] School of Mathematics and Computational Science,
Sun Yat-Sen University, Guangzhou, 510275, P.R. China
linxibin@mail2.sysu.edu.cn
[3] School of Computing, Dublin City University,
Ballymun, Dublin 9, Ireland
mike@computing.dcu.ie

Abstract. Efficiently computable homomorphisms allow elliptic curve point multiplication to be accelerated using the Gallant-Lambert-Vanstone (GLV) method. We extend results of Iijima, Matsuo, Chao and Tsujii which give such homomorphisms for a large class of elliptic curves by working over $\mathbb{F}_{p^2}$ and demonstrate that these results can be applied to the GLV method.

In general we expect our method to require about 0.75 the time of previous best methods (except for subfield curves, for which Frobenius expansions can be used). We give detailed implementation results which show that the method runs in between 0.70 and 0.84 the time of the previous best methods for elliptic curve point multiplication on general curves.

Keywords: elliptic curves, point multiplication, GLV method, isogenies.

1 Introduction

Let E be an elliptic curve over a finite field $\mathbb{F}_q$ and let $P, Q \in E(\mathbb{F}_q)$ have order r. The fundamental operations in elliptic curve cryptography are point multiplication $[n]P$ and multiexponentiation $[n]P+[m]Q$ where $n, m \in \mathbb{Z}$. There is a vast literature on efficient methods for computing $[n]P$ and $[n]P + [m]Q$ (a good reference is [3]). There is a significant difference between computing $[n]P$ for varying n and a fixed point P, and computing $[n]P$ where both n and P vary; this paper focusses on the latter case.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

⋆ This work supported by EPSRC grant EP/D069904/1.
⋆⋆ This author thanks the Chinese Scholarship Council.
⋆⋆⋆ This author acknowledges support from the Science Foundation Ireland under Grant No. 06/MI/006.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 518–535, 2009.

The Gallant-Lambert-Vanstone (GLV) method [15] is an important tool for speeding up point multiplication. The basic idea is as follows. If the elliptic curve E has an efficiently computable endomorphism ψ (other than a standard multiplication by n map) such that $\psi(P) \in \langle P \rangle$ then one can replace the computation $[n]P$ by the multiexponentiation $[n_0]P + [n_1]\psi(P)$ where $|n_0|, |n_1| \approx \sqrt{r}$. The integers n_0 and n_1 are computed by solving a closest vector problem in a lattice, see [15] for details. In principle this computation requires only around 0.6 to 0.7 the time of the previous method (the precise details depend on the relative costs of doubling and addition, the window size being used, etc). Some examples allow higher degree decompositions such as $[n_0] + [n_1]\psi(P) + \cdots + [n_{m-1}]\psi^{m-1}(P)$ where $|n_i| \approx r^{1/m}$ which can give further speedups. We call the latter approach the m-dimensional GLV method.

Gallant, Lambert and Vanstone [15] only gave examples of suitable efficiently computable endomorphisms in two cases, namely subfield curves (i.e., groups $E(\mathbb{F}_{q^m})$ where E is defined over $\mathbb{F}_q$; these do not have prime or nearly prime order unless q is very small) and curves with special endomorphism structure (essentially, that the endomorphism ring has small class number). Hence, if one is using randomly chosen prime-order elliptic curves over finite fields for cryptography (or if one wants to use special primes such as NIST primes, see Section 2.2.6 of [18]) then the GLV method is not usually available. Indeed, in Section 7 of [33] one finds the claim "the GLV method is only effective for those exceptional elliptic curves that have complex multiplication by an order with small discriminant."

In fact, Iijima, Matsuo, Chao and Tsujii [20] constructed an efficiently computable homomorphism on elliptic curves $E(\mathbb{F}_{p^2})$ with $j(E) \in \mathbb{F}_p$ arising from the Frobenius map on a twist of E. Apparently they did not realise the application of their results to the GLV method. In this paper we give a generalisation of the Iijima-Matsuo-Chao-Tsujii (IMCT) construction and analyse it in the context of the GLV method. The construction applies to all elliptic curves over $\mathbb{F}_{p^2}$ such that $j(E) \in \mathbb{F}_p$ and, as noted in [20,29], can be used with curves of prime order.

The curves considered in this paper are not completely general: the number of $\mathbb{F}_{q^2}$-isogeny classes of elliptic curves over $\mathbb{F}_{q^2}$ is approximately $2q^2$ whereas the construction in Section 2 gives only approximately q isomorphism classes of curves. However, this is a major improvement over earlier papers on the GLV method which, in practice, were only applied to a finite number of $\mathbb{F}_q$-isomorphism classes for any given q. The results of this paper therefore overturn the claims of Section 7 of [33].

The basic idea is somewhat analogous to subfield curves: We take elliptic curves E with $j(E) \in \mathbb{F}_q$ and consider the group $E(\mathbb{F}_{q^m})$. However a crucial difference is that E is defined over $\mathbb{F}_{q^m}$, not $\mathbb{F}_q$. This means that it is possible to obtain curves of prime order and so there is no need to restrict attention to q being small. Our method can be used with any prime power q and any elliptic curves E over $\mathbb{F}_q$ and always gives rise to a GLV method of dimension at least two.

We give experimental results comparing the cost of our algorithm for point multiplication $[n](x, y)$ with previous methods for this operation (indeed, we

compare with optimised implementations due to Bernstein [4] and Gaudry-Thomé [17], which, based on ideas of Montgomery [28], use only x-coordinate arithmetic). The purpose of our implementation experiments is to give a good picture of the speedup obtained with the new method compared with using curves over prime fields; we stress that our implementation is not claimed to be the best possible and that one could probably achieve further speedups from a different choice of curve coordinates or different exponentiation methods.

We find that the new method runs in between 0.70 and 0.84 the time of the previous best methods. The exact performance depends on the platform being used; our best result is for 8-bit processors. Our methods (unlike methods using Montgomery ladders, such as [4,17]) can also be used for signature verification. Our experimental results in Table 4 show that Schnorr signature verification runs in around 0.73 the time of the best previous methods for the same curve.

Note that our techniques can also be implemented on elliptic curves given by any equation (e.g., Edwards or Jacobi-quartic form, see [6,7,8]) and exploit their benefits. We also generalise the method to hyperelliptic curves. The details of both these cases are omitted due to lack of space, but are given in the full version of the paper.

The focus in this paper is on curves over fields of large prime characteristic, since in small characteristic one might prefer to use subfield curves and Frobenius expansions. However, Hankerson, Karabina and Menezes [19] have experimented with the method in characteristic 2 and they report that the new method runs in about 0.74 to 0.77 the time of the best standard method for general curves.

We now give an outline of the paper. First we describe the homomorphism and explain how it leads to a 2-dimensional GLV method. Section 3 gives a specific key generation algorithm which may be convenient for some applications. Section 4 shows how to get a 4-dimensional GLV method for $y^2 = x^3 + B$ over $\mathbb{F}_{p^2}$. Section 5 gives some details about our implementation. The proof of the pudding is the timings in Section 6. Section 7 discusses known security threats from using the construction and explains how to avoid them.

2 The Homomorphism

We consider elliptic curves defined over any field $\mathbb{F}_q$ with identity point $\mathcal{O}_E$. Recall that if E is an elliptic curve over $\mathbb{F}_q$ with $q + 1 - t$ points then one can compute the number of points $\#E(\mathbb{F}_{q^m})$ efficiently. For example, $\#E(\mathbb{F}_{q^2}) = q^2 + 1 - (t^2 - 2q) = (q + 1)^2 - t^2$. As usual we define

$$E(\mathbb{F}_{q^m})[r] = \{P \in E(\mathbb{F}_{q^m}) : [r]P = \mathcal{O}_E\}.$$

When we say that a curve or mapping is 'defined over $\mathbb{F}_{q^k}$' we mean that the coefficients of the polynomials are all in $\mathbb{F}_{q^k}$. The implicit assumption throughout the paper is that when we say an object is defined over a field $\mathbb{F}_{q^k}$ then it is not defined over any smaller field, unless explicitly mentioned.

The following result gives the main construction. Novices can replace the words 'separable isogeny' with 'isomorphism', set $d = 1$ and replace $\hat{\phi}$ by ϕ^{-1}

without any significant loss of functionality (in which case one essentially obtains the result of Iijima et al [20]). Recall that if r is a prime we write $r \| N$ to mean $r \mid N$ but $r^2 \nmid N$.

Theorem 1. *Let E be an elliptic curve defined over $\mathbb{F}_q$ such that $\#E(\mathbb{F}_q) = q + 1 - t$ and let $\phi : E \to E'$ be a separable isogeny of degree d defined over $\mathbb{F}_{q^k}$ where E' is an elliptic curve defined over $\mathbb{F}_{q^m}$ with $m \mid k$. Let $r \mid \#E'(\mathbb{F}_{q^m})$ be a prime such that $r > d$ and such that $r \| \#E'(\mathbb{F}_{q^k})$. Let π be the q-power Frobenius map on E and let $\hat{\phi} : E' \to E$ be the dual isogeny of ϕ. Define*

$$\psi = \phi\pi\hat{\phi}.$$

Then

1. *$\psi \in End_{\mathbb{F}_{q^k}}(E')$ (i.e., ψ is a group homomorphism).*
2. *For all $P \in E'(\mathbb{F}_{q^k})$ we have $\psi^k(P) - [d^k]P = \mathcal{O}_E$ and $\psi^2(P) - [dt]\psi(P) + [d^2q]P = \mathcal{O}_E$.*
3. *There is some $\lambda \in \mathbb{Z}$ such that $\lambda^k - d^k \equiv 0 \pmod{r}$ and $\lambda^2 - dt\lambda + d^2q \equiv 0 \pmod{r}$ such that $\psi(P) = [\lambda]P$ for all $P \in E'(\mathbb{F}_{q^m})[r]$.*

Proof. First note that $\hat{\phi}$ is an isogeny from E' to E and is defined over $\mathbb{F}_{q^k}$, that π is an isogeny from E to itself defined over $\mathbb{F}_q$, and that ϕ is an isogeny from E to E' defined over $\mathbb{F}_{q^k}$. Hence ψ is an isogeny of E' to itself, and is defined over $\mathbb{F}_{q^k}$ (or maybe a subfield). Therefore, ψ is a group homomorphism.

Since $\phi\hat{\phi} = d$ on E' it follows that

$$\psi^2 = \phi\pi\hat{\phi}\phi\pi\hat{\phi} = \phi\pi d\pi\hat{\phi} = d\phi\pi^2\hat{\phi}$$

and, by induction, $\psi^k = d^{k-1}\phi\pi^k\hat{\phi}$. For $P \in E'(\mathbb{F}_{q^k})$ we have $\hat{\phi}(P) \in E(\mathbb{F}_{q^k})$ and so $\pi^k(\hat{\phi}(P)) = \hat{\phi}(P)$. Hence $\psi^k(P) = [d^k]P$.

Similarly, writing $Q = \hat{\phi}(P)$ for $P \in E'(\mathbb{F}_{q^k})$ we have $\pi^2(Q) - [t]\pi(Q) + [q]Q = \mathcal{O}_E$ and so $[d]\phi(\pi^2 - [t]\pi + [q])\hat{\phi}(P) = \mathcal{O}_E$. Using the previous algebra, this implies

$$(\psi^2 - [dt]\psi + [qd^2])P = \mathcal{O}_E.$$

Finally, let $P \in E'(\mathbb{F}_{q^m})$ have order r. Since $\psi(P) \in E'(\mathbb{F}_{q^k})$ also has order r and $r \| \#E'(\mathbb{F}_{q^k})$ it follows that $\psi(P) = [\lambda]P$ for some $\lambda \in \mathbb{Z}$. Since ψ is a homomorphism, $\psi([a]P) = [a]\psi(P) = [\lambda]([a]P)$ for all $a \in \mathbb{Z}$. Since $\psi^k(P) - [d^k]P = [\lambda^k]P - [d^k]P = \mathcal{O}_E$ it follows that $\lambda^k - d^k \equiv 0 \pmod{r}$. Similarly, $\lambda^2 - dt\lambda + d^2q \equiv 0 \pmod{r}$. □

We stress that there is nothing unexpected in the above construction. Consider the case when ϕ is an isomorphism: Then $E' \cong E$ implies $\mathrm{End}(E') \cong \mathrm{End}(E)$. We know that $\mathrm{End}(E)$ contains the p-power Frobenius map and hence $\mathrm{End}(E')$ contains a corresponding endomorphism. The above Theorem simply writes down this endomorphism explicitly.

The proof generalises immediately to hyperelliptic curves (see the full version of this paper or [22]).

2.1 Special Case of Quadratic Twists

We now specialise Theorem 1 to elliptic curves over $\mathbb{F}_p$ where $p > 3$ and the case $m = 2$.

Corollary 1. *Let $p > 3$ be a prime and let E be an elliptic curve over $\mathbb{F}_p$ with $p + 1 - t$ points. Let E' over $\mathbb{F}_{p^2}$ be the quadratic twist of $E(\mathbb{F}_{p^2})$. Then $\#E'(\mathbb{F}_{p^2}) = (p-1)^2 + t^2$. Let $\phi : E \to E'$ be the twisting isomorphism defined over $\mathbb{F}_{p^4}$. Let $r \mid \#E'(\mathbb{F}_{p^2})$ be a prime such that $r > 2p$ Let $\psi = \phi\pi\phi^{-1}$. For $P \in E'(\mathbb{F}_{p^2})[r]$ we have $\psi^2(P) + P = \mathcal{O}_E$.*

Proof. Let $E : y^2 = x^3 + Ax + B$ with $A, B \in \mathbb{F}_p$. We have $\#E(\mathbb{F}_{p^2}) = p^2 + 1 - (t^2 - 2p)$. Let $u \in \mathbb{F}_{p^2}$ be a non-square in $\mathbb{F}_{p^2}$, define $A' = u^2A, B' = u^3B$ and $E' : y^2 = x^3 + A'x + B'$. Then E' is the quadratic twist of $E(\mathbb{F}_{p^2})$ and $\#E'(\mathbb{F}_{p^2}) = p^2 + 1 + (t^2 - 2p) = (p-1)^2 + t^2$. The isomorphism $\phi : E \to E'$ is given by

$$\phi(x, y) = (ux, \sqrt{u}^3 y)$$

and is defined over $\mathbb{F}_{p^4}$.

If $r \mid \#E'(\mathbb{F}_{p^2})$ is prime such that $r > 2p$ then $r \nmid \#E(\mathbb{F}_{p^2}) = (p+1-t)(p+1+t)$ and so $r \| \#E'(\mathbb{F}_{p^4}) = \#E(\mathbb{F}_{p^2})\#E'(\mathbb{F}_{p^2})$. Hence we may apply Theorem 1. This shows that $\psi = \phi\pi\phi^{-1}$ is a group homomorphism such that $\psi(P) = [\lambda]P$ for $P \in E'(\mathbb{F}_{p^2})[r]$ where $\lambda^4 - 1 \equiv 0 \pmod{r}$. We now show that, in fact, $\lambda^2 + 1 \equiv 0 \pmod{r}$.

By definition, $\psi(x, y) = (ux^p/u^p, \sqrt{u}^3 y^p / \sqrt{u}^{3p})$ where $u \in \mathbb{F}_{p^2}$ (i.e., $u^{p^2} = u$) and $\sqrt{u} \notin \mathbb{F}_{p^2}$ (and so, $\sqrt{u}^{p^2} = -\sqrt{u}$). If $P = (x, y) \in E'(\mathbb{F}_{p^2})$ then $x^{p^2} = x, y^{p^2} = y$ and so

$$\begin{aligned} \psi^2(x, y) &= (ux^{p^2}/u^{p^2}, \sqrt{u}^3 y^{p^2} / \sqrt{u}^{3p^2}) \\ &= (x, (-1)^3 y) \\ &= -(x, y). \end{aligned}$$

This completes the proof. □

The above result applies to any elliptic curve over $\mathbb{F}_p$ (with $p > 3$) and shows that the 2-dimensional GLV method can be applied. Note that it is possible for $\#E'(\mathbb{F}_{p^2})$ to be prime, since E' is not defined over $\mathbb{F}_p$ (for further analysis see Nogami and Morikawa [29]). One feature of this construction is that, since p is now half the size compared with using elliptic curves over prime fields, point counting is much faster than usual (this was noted in [29]). Since we are dealing with elliptic curves over $\mathbb{F}_{p^2}$, where p is prime, Weil descent attacks are not a threat (see Section 7).

An exercise for the reader is to show that if E is an elliptic curve over $\mathbb{F}_p$ and if E' over $\mathbb{F}_p$ is the quadratic twist of E then the map ψ satisfies $\psi(P) = -P$ for all $P \in E'(\mathbb{F}_p)$. The homomorphism is therefore useless for the GLV method in this case.

Lemma 1. *Let $p \equiv 5 \pmod 8$ be a prime. Let notation be as in Corollary 1. Then one may choose*

$$\psi(x, y) = (-x^p, iy^p)$$

where $i \in \mathbb{F}_p$ satisfies $i^2 = -1$.

Proof. See the full version of the paper. □

Lemma 2. *Let notation be as in Corollary 1. Then $\psi(P) = [\lambda]P$ where $\lambda = t^{-1}(p-1) \pmod r$.*

Proof. The proof of Corollary 1 shows that $\psi(P) = [\lambda]P$ for some $\lambda \in \mathbb{Z}$. Since $\psi^2(P) = -P$ we have $\lambda^2+1 \equiv 0 \pmod r$. Similarly, $\psi^2(P)-[t]\psi(P)+[p]P = \mathcal{O}_E$, so $\lambda^2 - t\lambda + p \equiv 0 \pmod r$. Subtracting the second equation from the first gives $t\lambda + (1-p) \equiv 0 \pmod r$. □

Finally, we give some remarks about the lattice which arises in the GLV method when decomposing $[n]P$ as $[n_0]P + [n_1]\psi(P)$. Recall from [15] that we consider the lattice

$$L = \{(x, y) \in \mathbb{Z}^2 : x + y\lambda \equiv 0 \pmod r\}.$$

It is easy to prove that $\{(r, 0), (-\lambda, 1)\}$ is a basis for L; this shows that the determinant of L is r. The GLV method uses Babai's rounding method to solve the closest vector problem (CVP), and this method requires a reduced basis.

Lemma 3. *Let notation be as in Corollary 1. The vectors $\{(t, p-1), (1-p, t)\}$ are an orthogonal basis for a sublattice L' of L of determinant $\#E'(\mathbb{F}_{p^2})$. Given a point $(a, b) \in \mathbb{R}^2$ there exists a lattice point $(x, y) \in L'$ such that $\|(a, b)-(x, y)\| \leq (p+1)/\sqrt{2}$.*

Proof. By Lemma 2 we have that $t\lambda + (1-p) \equiv 0 \pmod r$, which proves that $(1-p, t) \in L$. Multiplying by λ and using $\lambda^2 \equiv -1 \pmod r$ gives $(t, p-1) \in L$. It is easy to check that the vectors are orthogonal and thus linearly independent. The vectors both have length $\sqrt{\#E'(\mathbb{F}_{p^2})} \leq \sqrt{p^2+2p+1} = p+1$. This basis has determinant $(p-1)^2 + t^2 = \#E'(\mathbb{F}_{p^2})$ so generates a sublattice $L' \subseteq L$ (if $\#E'(\mathbb{F}_{p^2}) = r$ then $L = L'$).

Finally, simple geometry shows that the maximum distance from a lattice point is $\sqrt{\#E'(\mathbb{F}_{p^2})/2} \leq (p+1)/\sqrt{2}$. □

Computing the coefficients n_0, n_1 for the GLV method is therefore particularly simple in this case (one does not need to use lattice reduction or the methods of [30,21,33]). Further, one knows that $|n_0|, |n_1| \leq (p+1)/\sqrt{2}$. As always, an alternative to the decomposition method which can be used in some cryptographic settings is to choose small coefficients $n_0, n_1 \in \mathbb{Z}$ directly rather than choosing a random $0 \leq n < r$ and then computing the corresponding (n_0, n_1).

2.2 Higher Dimension Decompositions

The GLV method can be generalised to m-dimensional decompositions $[n]P = [n_0]P + [n_1]\psi(P) + \cdots + [n_{m-1}]\psi^{m-1}(P)$ (for examples with $m = 4$ and $m = 8$ see [13]). Such a setting gives improved performance. As we have found 2-dimensional expansions using $E'(\mathbb{F}_{p^2})$ it is natural to try to get an m-dimensional decomposition using $E'(\mathbb{F}_{p^m})$.

In general, to obtain an m-dimensional decomposition it is required that ψ does not satisfy any polynomial equation on $E'(\mathbb{F}_{p^m})[r]$ of degree $< m$ with small integer coefficients. Note that ψ always satisfies a quadratic polynomial equation but that the coefficients are not necessarily small modulo r.

The following result gives a partial explanation of the behaviour of ψ on $E'(\mathbb{F}_{p^m})$.

Corollary 2. *Let $p > 3$ be a prime and let E be an elliptic curve over $\mathbb{F}_p$. Let E' over $\mathbb{F}_{p^m}$ be the quadratic twist of $E(\mathbb{F}_{p^m})$. Write $\phi : E \to E'$ for the twisting isomorphism defined over $\mathbb{F}_{p^{2m}}$. Let $r \mid \#E'(\mathbb{F}_{p^m})$ be a prime such that $r > 2p^{m-1}$ Let $\psi = \phi\pi\phi^{-1}$. For $P \in E'(\mathbb{F}_{p^m})[r]$ we have $\psi^m(P) + P = \mathcal{O}_E$.*

Proof. As in Corollary 1, we have $r \| \#E'(\mathbb{F}_{p^{2m}}) = \#E'(\mathbb{F}_{p^m})\#E(\mathbb{F}_{p^m})$ so Theorem 1 applies. Using the same method as the proof of Corollary 1 we have $\psi^m(x, y) = (ux^{p^m}/u^{p^m}, \sqrt{u}^3 y^{p^m}/\sqrt{u}^{3p^m}) = -P$. □

A problem is that the polynomial $x^m + 1$ is not usually irreducible, and it is possible that ψ satisfies a smaller degree polynomial. For example, in the case $m = 3$ one sees that $\#E'(\mathbb{F}_{p^3})$ cannot be prime as it is divisible by $N = \#E(\mathbb{F}_{p^2})/\#E(\mathbb{F}_p)$. If $r \mid \#E'(\mathbb{F}_{p^3})/N$ and $P \in E'(\mathbb{F}_{p^3})[r]$ then $\psi^2(P) - \psi(P) + 1 = \mathcal{O}_E$. Hence one only gets a 2-dimensional decomposition in the case $m = 3$.

Indeed, the interesting case is when m is a power of 2, in which case $x^m + 1$ is irreducible and one can obtain an m-dimensional GLV decomposition. Indeed, Nogami and Morikawa [29] already proposed exactly this key generation method (choosing E over $\mathbb{F}_p$ and then using a quadratic twist over $\mathbb{F}_{p^{2^c}}$) as a method to generate curves of prime order. Note that [29] does not consider the GLV method.

Therefore, the next useful case is $m = 4$, giving a 4-dimensional GLV method. On the downside, this case is potentially vulnerable to Weil descent attacks (see Section 7) and so the prime p must be larger than we would ideally like.

The other way to get higher dimension decompositions is to have maps ϕ defined over larger fields than a quadratic extension. An example of this is given in Section 4.

3 Key Generation

Let $p > 3$ be prime. We present a key generation algorithm for the quadratic twist construction. Our algorithm is designed so that the resulting curve

$E' : y^2 = x^3 + A'x + B'$ over $\mathbb{F}_{p^2}$ has coefficient $A' = -3$, which is convenient for efficient implementation when using Jacobian coordinates (see Section 13.2.1.c of [3] or Section 3.2.2 of [18]). The key generation algorithm can be modified to work with other models for elliptic curves and one can always choose at least one coefficient to have a special form.

We use Lemma 1, which gives a particularly simple map ψ. It should be clear that the algorithm can be used in more general cases. Our algorithm produces curves of prime order, but this can be relaxed by requiring only $h < H$ for some bound H in line 7.

Algorithm 1. Key generation for quadratic twist construction

OUTPUT: p, E', ψ, λ
1: Choose a prime $p = 5 \pmod 8$ ▷ e.g., a NIST prime (Section 2.2.6 of [18])
2: Set $u = \sqrt{2} \in \mathbb{F}_{p^2}$
3: Set $A' = -3$ and $A = A'/2 \in \mathbb{F}_p$
4: **repeat**
5: Choose random $B \in \mathbb{F}_p$ and let $E : y^2 = x^3 + Ax + B$
6: Compute $t = p + 1 - \#E(\mathbb{F}_p)$.
7: **until** $(p-1)^2 + t^2 = hr$ where r is prime and $h = 1$
8: Set $B' = Bu^3 \in \mathbb{F}_{p^2}$ and $E' : y^2 = x^3 + A'x + B'$
9: Set $\lambda = t^{-1}(p-1) \pmod r$
10: Compute $i \in \mathbb{F}_p$ so that $i^2 = -1$
11: Define $\psi(x, y) = (-x^p, iy^p)$.
12: **return** $p, (A', B'), \psi, \lambda$

As remarked earlier, key generation is fast compared with standard ECC, since the point counting for $\#E(\mathbb{F}_p)$ is over a field half the usual size (this is precisely the point of the paper [29]).

4 Using Special Curves

We have seen that one can obtain a 2-dimensional GLV method for any elliptic curve over $\mathbb{F}_p$. However, 2-dimensional GLV methods were already known for some special curves (i.e., those with a non-trivial automorphism or endomorphism of low degree). We now show how one can get higher-dimensional expansions using elliptic curves E over $\mathbb{F}_{p^2}$ with $\#\mathrm{Aut}(E) > 2$.

The two examples of interest are $E : y^2 = x^3 + B$ and $y^2 = x^3 + Ax$. We give the details in the former case. The latter is analogous.

Let $p \equiv 1 \pmod 6$ and let $B \in \mathbb{F}_p$. Define $E : y^2 = x^3 + B$. Choose $u \in \mathbb{F}_{p^{12}}$ such that $u^6 \in \mathbb{F}_{p^2}$ and define $E' : Y^2 = X^3 + u^6 B$ over $\mathbb{F}_{p^2}$. Repeat the construction (choosing p, B, u) until $\#E'(\mathbb{F}_{p^2})$ is prime (or nearly prime). Note that there are 6 possible group orders for $y^2 = x^3 + B'$ over $\mathbb{F}_{p^2}$ and three of them are never prime as they correspond to group orders of curves defined over $\mathbb{F}_p$.

The isomorphism $\phi : E \to E'$ is given by $\phi(x, y) = (u^2 x, u^3 y)$ and is defined over $\mathbb{F}_{p^{12}}$. The homomorphism $\psi = \phi\pi\phi^{-1}$, where π is the p-power Frobenius on E, is defined over $\mathbb{F}_{p^2}$ and satisfies the characteristic equation

$$\psi^4 - \psi^2 + 1 = 0$$

corresponding to the 12-th cyclotomic polynomial. Hence one obtains a 4-dimensional GLV method for these curves. This leads, once again, to a significant speedup of these curves compared with previous techniques.

Note that $-\psi^2$ satisfies the characteristic equation $x^2 + x + 1$ and so acts as the standard automorphism $(x, y) \mapsto (\zeta_3 x, y)$ on E.

5 Remarks on Our Implementation

In this section we briefly describe the implementation we used for our experiments. As mentioned in the introduction, we do not claim that our implementation is the best possible. We believe that, for the parameters and implementation platforms considered in this paper, it gives a fair estimate of the speedup obtained by using the GLV method.

The main point of the GLV method is to replace a large point multiplication $[n]P$ by a multiexponentiation $[n_0]P + [n_1]\psi(P)$. There are numerous algorithms for multiexponentiation, all built on a fundamental observation by Straus, and much has been written on the topic. One approach is to use 'interleaving'; this idea seems to have been independently discovered in [15] and [24]. We refer to Section 3.3.3 of [18] for details. Another approach is the joint sparse form (see Solinas [34]). The full version of the paper contains further analysis of multiexponentiation methods (e.g., higher-dimensional joint sparse forms, the Euclidean Montgomery ladder etc).

Two fundamental ideas used to speed up the computation of $[n]P$ on elliptic curves are the use of signed binary expansions (for example, non-adjacent forms, see Definition 3.28 [18] or Definition 9.13 of [3]) and sliding window methods. A very efficient method (as it only uses a few word operations) to compute the NAF of an integer n is to compute $3n$ (using standard integer multiplication), then form the signed expansion $(3n) - n$ and discard the least significant bit. The natural extension of non-adjacent forms to windows is called width-w NAFs (see Section IV.2.5 of [9], Definition 3.32 of [18] or Definition 9.19 of [3]). Instead of using width-w NAFs one can use sliding windows over NAF expansions (see Section IV.2.4 of [9] or Algorithm 3.38 on page 101 of [18]). This is convenient since it is cheaper to compute a NAF than a width-w NAF.

More generally, one can use signed fractional windows [25,26]. Finally, one could consider fractional sliding windows over NAFs. This does not seem to have been considered in the literature and it is an open problem to determine the density in this case. More details of these methods are given in the full version of the paper.

Our implementation uses interleaving with sliding (non-fractional) windows of width $w = 4$ over NAF expansions (we found that using $w = 5$ was slightly

slower for our parameters). Hence we must precompute $\{P, [3]P, [5]P, [7]P, [9]P\}$; note that the points $\{\psi(P), [3]\psi(P), [5]\psi(P), [7]\psi(P), [9]\psi(P)\}$ can be obtained on the fly at little cost. This is very similar to Algorithm 3.51 of [18], which uses interleaving over width-w NAFs (the authors of [18] tell us that there is a typo in line 2 of Algorithm 3.5.1: one should replace "3.30" with "3.35"). We do not claim that this is the fastest possible approach, but it requires relatively little precomputation and is very simple to implement. It is possible that one could obtain slightly faster results using fractional windows or other methods.

The next decision is which coordinate system to use for elliptic curve arithmetic. The best choice is probably inverted Edwards or Jacobi quartic [6,7,8] but for legacy reasons our implementation uses Jacobian coordinates. As usual, one prefers to use mixed additions in the main loop as they are faster. However this requires that any precomputed values must be "normalized", that is converted to affine form, before entering the loop. This conversion, if done naively for each precomputed point, would require expensive field inversions, so we use the precomputation strategy of Dahmen, Okeya and Schepers (DOS) [12], as recommended in [8] (there are also recent improvements due to Longa and Miri [23]), which requires only a single inversion.

The full version of the paper gives more details of the implementation, as well as a theoretical estimate of the number of $\mathbb{F}_{p^2}$ operations required for our algorithm.

6 Experimental Results

We now give some timing comparisons for the computation of $[n]P$ (and also signature verification) on elliptic curves at the 128-bit security level. Our timings are for the case of quadratic twists as presented in Section 2.1.

6.1 The Example Curve

It is natural to use the Mersenne prime $p = 2^{127} - 1$, which is also used in Bernstein's `surface1271` genus 2 implementation [5][1]. This prime supports a very fast modular reduction algorithm.

Since $p \equiv 3 \pmod 4$ we represent $\mathbb{F}_{p^2}$ as $\mathbb{F}_p(\sqrt{-1})$. Note that since $p \neq 5 \bmod 8$ the previously described key generation process is not applicable here. However it can easily be modified to handle this case as well, although the homomorphism requires more multiplications to compute.

Let

$$E : y^2 = x^3 - 3x + 44$$

[1] Note that the Pollard rho algorithm using equivalence classes in this case requires approximately 2^{125} group operations, the same as for Bernstein's Curve25519 or Surface1271. Whether this is precisely the same security level as AES-128 is unclear, but since Curve25519 and Surface1271 have been used for benchmarking we feel our choice is justified.

be defined over the field $\mathbb{F}_p$. Then $\#E(\mathbb{F}_p) = p+1-t$ where $t =$ `3204F5AE088C39A7` in hex. By Corollary 1 the quadratic twist E' over $\mathbb{F}_{p^2}$ of $E(\mathbb{F}_{p^2})$ has $\#E'(\mathbb{F}_{p^2}) = (p-1)^2+t^2$, which is a prime we call r. The curve was quickly found using a modified version of Schoof's algorithm.

We use $u = 2 + i$ instead of $u = \sqrt{2}$ in Algorithm 1. The homomorphism in this case simplifies to

$$\psi(x, y) = (\omega_x \bar{x}, \omega_y \bar{y})$$

where $\bar{x}$ denotes the Galois conjugate of x, and $\omega_x = u/u^p, \omega_y = \sqrt{u^3/u^{3p}}$ as in the proof of Corollary 1. By Lemma 2 we have $\psi(P) = [\lambda]P$ where $\lambda = t^{-1}(p-1) \pmod{r}$.

6.2 Comparison Curve

For comparison purposes we consider an elliptic curve E defined over $\mathbb{F}_{p_2}$ where $p_2 = 2^{256} - 189$ is a 256-bit pseudo-Mersenne modulus. This provides approximately the same level of security as the curve in the previous subsection.

The full version of the paper gives a theoretical comparison of the implementations. Table 1 gives operation counts for our test implementation. The notation SSW means sliding windows of window size $w = 5$ over NAFs, GLV+JSF means using joint sparse forms for the multiexponentiation and GLV+INT means interleaving sliding windows of size 4 over NAFs as described in Section 5. In our implementations we averaged the cost over 10^5 point multiplications.

Table 1. Point multiplication operation counts

	Method	$\mathbb{F}_p$ muls	$\mathbb{F}_p$ adds/subs
$E(\mathbb{F}_{p_2})$, 256-bit p_2	SSW	2600	3775
$E(\mathbb{F}_{p^2})$, 127-bit p	SSW	6641	16997
$E(\mathbb{F}_{p^2})$, 127-bit p	GLV+JSF	4423	10785
$E(\mathbb{F}_{p^2})$, 127-bit p	GLV+INT	4109	10112

The results in Table 1 agree with the rough analysis given in the full version of the paper. The table includes the often neglected costs of field additions and subtractions. Note that when implementing $\mathbb{F}_{p^2}$ arithmetic, each multiplication using Karatsuba requires five $\mathbb{F}_p$ additions or subtractions (assuming $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$), so the number of these operations increases substantially.

Clearly the superiority (or otherwise) of the method depends on the relative cost of 128-bit and 256-bit field multiplications (and additions or subtractions) on the particular platform.

To give a more accurate picture we have implemented both methods on two widely differing platforms, a 1.66GHz 64-bit Intel Core 2, and on an 8-bit 4MHz Atmel Atmega1281 chip (which is a popular choice for wireless sensor network nodes). We present the results in the following two subsections.

6.3 8-bit Processor Implementation

Our first implementation is on a small 4MHz 8-bit Atmega1281 processor. Here the base field multiplication times will dominate, so this function was written in optimal loop-unrolled assembly language. We use the MIRACL C library [31], which includes tools for the automatic generation of such code (and which holds the current speed record for this particular processor [32]), and we use the cycle accurate AVR Studio tool to measure the time for a single variable point multiplication.

Table 2. Point multiplication timings – 8-bit processor

Atmel Atmega1281 processor	Method	Time (s)
$E(\mathbb{F}_{p_2})$, (256-bit p_2)	SSW	5.49
$E(\mathbb{F}_{p^2})$ (127-bit p)	SSW	6.20
$E(\mathbb{F}_{p^2})$, (127-bit p)	GLV+JSF	4.21
$E(\mathbb{F}_{p^2})$, (127-bit p)	GLV+INT	3.87

Table 2 show a that our best method for point multiplication takes about 0.70 of the time required for the 256 bit $E(\mathbb{F}_{p_2})$ curve.

Observe that simply switching to an $E(\mathbb{F}_{p^2})$ curve at the same security level does not by itself give any improvement, in fact it is somewhat slower. The theoretical advantage of using Karatsuba in the latter case appears to be outweighed by the extra "fussiness" of the $\mathbb{F}_{p^2}$ implementation; and of course Karatsuba can also be applied to the $\mathbb{F}_p$ case as well if considered appropriate. Looking at the timings, a field multiplication takes 1995 μs over $\mathbb{F}_{p_2}$ (256-bit), as against 2327 μs over $\mathbb{F}_{p^2}$ (127-bit p), although for a field squaring the situation is reversed, taking 1616 μs over $\mathbb{F}_{p_2}$ as against only 1529 μs over $\mathbb{F}_{p^2}$. Field addition and subtraction favours the $\mathbb{F}_{p_2}$ case (124 μs versus 174 μs). However using the new homomorphism and applying the GLV method, our new implementation is still clearly superior.

Note that for this processor it is probably more appropriate in practice to use the JSF method for point multiplication, as it is much better suited to a small constrained enviroment, with limited space for online precomputation.

6.4 64-Bit Processor Implementation

It has been observed by Avanzi [2], that software implementations over smaller prime fields, where field elements can be stored in just a few CPU registers (as will be the case here), suffer disproportionally when implemented using general purpose multi-precision libraries. This effect would work against us here, as we are using the general purpose MIRACL library [31]. Special purpose libraries like the `mpFq` library [17] which generate field-specific code, and implementations which work hard to squeeze out overheads, such as Bernstein's implementations [5] are always going to be faster.

In the context of a 64-bit processor, while one might hope that timings would be dominated by the $O(n^2)$ base field multiplication operations, for small values of n the $O(n)$ contribution of the numerous base field additions and subtractions becomes significant, as also observed by Gaudry and Thomé [17]. Observe that on the 64-bit processor a 128-bit field element requires just $n = 2$ (and indeed the description as "multi-precision" should really give way to "double precision"). Therefore it is to be expected that the speed-up we can achieve in this case will be less than might have been hoped.

So is our new method faster? There is really only one satisfactory way to resolve the issue – and that is to identify the fastest known $E(\mathbb{F}_{p_2})$ implementation on a 64-bit processor for the same level of security, and try to improve on it. We understand that the current record is that announced by Gaudry and Thomé at SPEED 2007 [17], using an implementation of Bernstein's `curve25519` [4]. This record is in the setting of an implementation of the elliptic curve Diffie-Hellman method, which requires a single point multiplication to determine the shared secret key.

We point out that the clever implementation and optimizations of `curve25519` are for the sole context of an efficient Diffie-Hellman implementation – ours is general purpose and immediately applicable to a wide range of ECC protocols. In particular the implementation of `curve25519` uses Montgomery's parameterisation of an elliptic curve, is not required to maintain a y coordinate, and hence can achieve compression of the public key at no extra cost (i.e., without the calculation of a square root).

On the other hand we have the use of a particularly nice modulus $2^{127} - 1$, which brings many benefits. For example a base field square root of a quadratic residue x can be calculated as simply $x^{2^{125}}$.

In order to be competitive we wrote a specialised hand-crafted x86-64 assembly language module to handle the base field arithmetic, and integrated this with the MIRACL library. Given that each field element can be stored in just two 64-bit registers, this code is quite short, and did not take long to generate, optimize and test.

To obtain our timings we follow Gaudry and Thomé, and utilise two different methods, one based on actual cycle counts, and a method which uses an operating system timer. There are problems with both methods [17], so here we average the two. In practise the two methods were in close agreement, but not of sufficient accuracy to justify exact numbers – so we round to the nearest 1000 cycles. See

Table 3. Point multiplication timings – 64-bit processor

Intel Core 2 processor	Method	Clock cycles
$E(\mathbb{F}_{p_2})$, 255-bit p_2	Montgomery [17]	386,000
$E(\mathbb{F}_{p^2})$, 127-bit p	SSW	490,000
$E(\mathbb{F}_{p^2})$, 127-bit p	GLV+JSF	359,000
$E(\mathbb{F}_{p^2})$, 127-bit p	GLV+INT	326,000

Table 3 for our results. As can be seen, our best method takes 0.84 of the time of the Gaudry and Thomé implementation. Note that point decompression, as required by a Diffie-Hellman implementation which wishes to minimise the size of the public key, would require approximately an extra 26,000 clock cycles for our implementation.

It is interesting to observe from Table 3 that a careful implementation over a quadratic extension which does not exploit our homomorphism is substantially slower, taking 490,000 cycles. So again it seems that merely switching to a smaller field size is not by itself advantageous on a 64-bit processor, although some of the difference can be explained by the particularly clever parameterization chosen for `curve25519`. However by using the GLV method we are able to make up this difference, and indeed overtake the previous record.

To ensure a fair comparison, we exploited the very useful `eBats` project [10] (now incorporated into eBACS [11]). Our eBat implements a Diffie-Hellman key exchange algorithm, and can be directly and independently compared with an implementation based on `curve25519`. There are two main functions for a Diffie-Hellman implementation, one which calculates the key pair, and a second which calculates the shared secret. For the key pair calculation we exploit the fact that for our method a multiplication of a fixed point can benefit from extensive off-line precomputation, and use a fixed-base comb algorithm (see Section 3.3.2 of [18]), and so this calculation requires only 146,000 cycles. For the shared secret calculation we use the GLV+INT method, plus the cost of a point decompression.

Our latest eBat can be downloaded from:

`ftp://ftp.computing.dcu.ie/pub/crypto/gls1271-3.tar`

Profiling the code reveals that our version (with point compression) spends 49% of its time doing base field multiplications and squarings, 15% of the time doing base field additions and subtractions and nearly 6% of the time is required for the few modular inversions.

6.5 ECDSA/Schnorr Signature Verification

Verification of both ECDSA and Schnorr signatures requires the calculation of $[a]P+[b]Q$, where P is fixed. In our setting we must calculate $[a_0]P+[a_1]\psi(P)+[b_0]Q+[b_1]\psi(Q)$ – in other words a 4-dimensional multiexponentiation algorithm is required. The methods of Bernstein [4] and Gaudry-Thomé [17] are based on Montgomery arithmetic and are not appropriate for signature verification.

Again we use an interleaving algorithm, using windows over a NAF expansion. Since P is now fixed, precomputation of multiples of P (and therefore of $\psi(P)$) can be carried out offline, and so a larger window size of 6 can be used for the multiplication of P. This requires the precomputation and storage of 42 points. For the online precomputation required on Q, we again use sliding windows of size 4 over NAF expansions.

In Table 4 we compare our method with an implementation that does not use the GLV method. The notation GLV+INT means a 4-dimensional multiexponentiation as described above and the notation INT means the 2-dimensional interleaving algorithm which calculates $[a]P+[b]Q$ directly for random $a, b < r$,

Table 4. Signature Verification timings – 64-bit processor

Intel Core 2 processor	Method	$\mathbb{F}_p$ muls	$\mathbb{F}_p$ adds/subs	Clock cycles
$E(\mathbb{F}_{p^2})$, 127-bit p	GLV+INT	5174	12352	425,000
$E(\mathbb{F}_{p^2})$, 127-bit p	INT	7638	19046	581,000

using size 6 sliding windows over NAFs for the fixed point P, and size 5 sliding windows over NAFs for the variable point Q.

Antipa et al [1] propose a variant of ECDSA with faster signature verification (note that their method does not apply to Schnorr signatures). The basic method gives essentially the same performance as our method (they transform $[a]P+[b]Q$ to a 4-dimensional multiexponentiation with coefficients $\approx \sqrt{r}$). Their method, as with ours, assumes that P is fixed and that certain precomputation has been done.

The paper [1] also gives a variant where the public key is doubled in size to include Q and $Q_1 = [2^{\lceil \log_2(r)/3 \rceil}]Q$. Their method transforms $[a]P + [b]Q$ to a 6-dimensional multiexponentiation with coefficients of size $\approx r^{1/3}$. In this context (i.e., enlarged public keys) we can improve upon their result. Let $M = 2^{\lceil \log_2(r)/4 \rceil}$ and suppose the public key features Q and $Q_1 = [M]Q$. The GLV idea transforms $[a]P+[b]Q$ to $[a_0]P+[a_1]\psi(P)+[b_0]Q+[b_1]\psi(Q)$ where $a_0, a_1, b_0, b_1 \approx \sqrt{r}$. We now write $a_0 = a_{0,0} + Ma_{0,1}$ where $a_{0,0}, a_{0,1} \approx r^{1/4}$ and similarly for a_1, b_0, b_1. Hence the computation becomes an 8-dimensional multiexponentiation with coefficients of size $\approx r^{1/4}$. Another advantage of our method is that it applies to Schnorr signatures whereas the method of [1] is only for ECDSA and other variants of ElGamal signatures.

Finally, we mention that the methods in [27] can also be applied in our setting.

7 Security Implications

The homomorphism ψ of Theorem 1 (at least, in the case when ϕ is an isomorphism) defines equivalence classes of points in $E'(\mathbb{F}_{p^m})$ of size $2m$ by $[P] = \{\pm\psi^i(P) : 0 \leq i < m\}$. By the methods of Gallant-Lambert-Vanstone [14] and Wiener-Zuccherato [35] one can perform the Pollard rho algorithm for the discrete logarithm problem on these equivalence classes. This speeds up the solution of the discrete logarithm problem by a factor of $\sqrt{m}$ compared with general curves. Hence one bit should be added to the key length to compensate for this attack.

A more serious threat comes from the Weil descent philosophy, and in particular the work of Gaudry [16]. Gaudry gives an algorithm for the discrete logarithm problem in $E'(\mathbb{F}_{p^m})$ requiring time $O(p^{2-4/(2m+1)})$ group operations (with bad constants) which, in principle, beats the Pollard methods for $m \geq 3$. The proposal for elliptic curves in the case $m = 2$ is immune to Gaudry's Weil descent attack.

Gaudry's method also applies to abelian varieties: if A is an abelian varitey of dimension d over $\mathbb{F}_{p^m}$ then the algorithm has complexity $O(p^{2-4/(2dm+1)})$. Hence, for Jacobians of genus 2 curves over $\mathbb{F}_{p^2}$ one has an algorithm running in time $O(p^{1.55})$, rather than the Pollard complexity of $O(p^2)$. Gaudry's method is exponential time and so one can secure against it by increasing the field size. For example, to achieve 128-bit security level with genus 2 curves over $\mathbb{F}_{p^2}$ or elliptic curves over $\mathbb{F}_{p^4}$ one should take p to be approximately 80 bits rather than the desired 64 bits (this is a very conservative choice; Gaudry's algorithm requries expensive computations such as Gröbner bases and so one can probably safely work with primes smaller than 80 bits).

Acknowledgements

We thank Dan Bernstein, Billy Brumley, Jinhui Chao, Pierrick Gaudry, Darrel Hankerson, Alfred Menezes, Yasuyuki Nogami, Fre Vercauteren and the anonymous referees for suggestions and comments.

References

1. Antipa, A., Brown, D., Gallant, R.P., Lambert, R., Struik, R., Vanstone, S.A.: Accelerated Verification of ECDSA Signatures. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 307–318. Springer, Heidelberg (2006)
2. Avanzi, R.M.: Aspects of Hyperelliptic Curves over Large Prime Fields in Software Implementations. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 148–162. Springer, Heidelberg (2004)
3. Avanzi, R., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Cryptography. Chapman and Hall/CRC (2006)
4. Bernstein, D.J.: Curve25519: New Diffie-Hellman Speed Records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
5. Bernstein, D.J.: Elliptic vs. Hyperelliptic, part 1 ECC, Toronto, Canada (2006), `http://www.cacr.math.uwaterloo.ca/conferences/2006/ecc2006/slides.html`
6. Bernstein, D.J., Lange, T.: Faster Addition and Doubling on Elliptic Curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
7. Bernstein, D.J., Lange, T.: Inverted Edwards Coordinates. In: Boztaş, S., Lu, H.-F. (eds.) AAECC 2007. LNCS, vol. 4851, pp. 20–27. Springer, Heidelberg (2007)
8. Bernstein, D.J., Lange, T.: Analysis and Optimization of Elliptic-Curve Single-Scalar Multiplication. In: Finite Fields and Applications: Proceedings of Fq8, Contemporary Mathematics 461, pp. 1–18. American Mathematical Society (2008)
9. Blake, I., Seroussi, G., Smart, N.P. (eds.): Elliptic Curves in Cryptography. Cambridge University Press, Cambridge (1999)
10. eBATS: ECRYPT Benchmarking of Asymmetric Systems, `http://www.ecrypt.eu.org/ebats/`
11. Bernstein, D.J., Lange, T.: eBACS: ECRYPT Benchmarking of Cryptographic Systems (accessed January 9, 2009), `http://bench.cr.yp.to/`

12. Dahmen, E., Okeya, K., Schepers, D.: Affine Precomputation with Sole Inversion in Elliptic Curve Cryptography. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 245–258. Springer, Heidelberg (2007)
13. Galbraith, S.D., Scott, M.: Exponentiation in Pairing-Friendly Groups Using Homomorphisms. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 211–224. Springer, Heidelberg (2008)
14. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Improving the Parallelized Pollard Lambda Search on Anomalous Binary Curves. Math. Comp. 69, 1699–1705 (2000)
15. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (2001)
16. Gaudry, P.: Index Calculus for Abelian Varieties of Small Dimension and the Elliptic Curve Discrete Logarithm Problem. J. Symbolic Comput. (to appear)
17. Gaudry, P., Thome, E.: The mpFq Library and Implementing Curve-Based Key Exchanges. In: SPEED workshop presentation, Amsterdam (June 2007)
18. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to elliptic curve cryptography. Springer, Heidelberg (2004)
19. Hankerson, D., Karabina, K., Menezes, A.J.: Analyzing the Galbraith-Lin-Scott Point Multiplication Method for Elliptic Curves over Binary Fields, eprint 2008/334
20. Iijima, T., Matsuo, K., Chao, J., Tsujii, S.: Costruction of Frobenius Maps of Twist Elliptic Curves and its Application to Elliptic Scalar Multiplication. In: SCIS 2002, IEICE Japan, pp. 699–702 (January 2002)
21. Kim, D., Lim, S.: Integer Decomposition for Fast Scalar Multiplication on Elliptic Curves. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 13–20. Springer, Heidelberg (2003)
22. Kozaki, S., Matsuo, K., Shimbara, Y.: Skew-Frobenius Maps on Hyperelliptic Curves, IEICE Trans. E91-A(7), 1839–1843 (2008)
23. Longa, P., Miri, A.: New Composite Operations and Precomputation Scheme for Elliptic Curve Cryptosystems over Prime Fields. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 229–247. Springer, Heidelberg (2008)
24. Möller, B.: Algorithms for Multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
25. Möller, B.: Improved Techniques for Fast Exponentiation. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
26. Möller, B.: Fractional Windows Revisited: Improved Signed-Digit Representations for Efficient Exponentiation. In: Park, C.-S., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 137–153. Springer, Heidelberg (2005)
27. Möller, B., Rupp, A.: Faster Multi-exponentiation through Caching: Accelerating (EC)DSA Signature Verification. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 39–56. Springer, Heidelberg (2008)
28. Montgomery, P.L.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Math. Comp. 47, 243–264 (1987)
29. Nogami, Y., Morikawa, Y.: Fast Generation of Elliptic Curves with Prime Order over Extension Field of Even Extension Degree. In: Proceedings 2003 IEEE International Symposium on Information Theory, p. 18 (2003)
30. Park, Y.-H., Jeong, S., Kim, C.-H., Lim, J.-I.: An Alternate Decomposition of an Integer for Faster Point Multiplication on Certain Elliptic Curves. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 323–334. Springer, Heidelberg (2002)

31. Scott, M.: MIRACL – Multiprecision Integer and Rational Arithmetic C/C++ Library (2008), http://ftp.computing.dcu.ie/pub/crypto/miracl.zip
32. Scott, M., Szczechowiak, P.: Optimizing Multiprecision Multiplication for Public Key Cryptography (2007), http://eprint.iacr.org/2007/299
33. Sica, F., Ciet, M., Quisquater, J.-J.: Analysis of the Gallant-Lambert-Vanstone Method based on Efficient Endomorphisms: Elliptic and Hyperelliptic Curves. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 21–36. Springer, Heidelberg (2003)
34. Solinas, J.A.: Low-Weight Binary Representations for Pairs of Integers, Technical Report CORR 2001–41, CACR (2001)
35. Wiener, M., Zuccherato, R.J.: Faster Attacks on Elliptic Curve Cryptosystems. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 190–200. Springer, Heidelberg (1999)

Generating Genus Two Hyperelliptic Curves over Large Characteristic Finite Fields

Takakazu Satoh*

Department of Mathematics,
Tokyo Institute of Technology, Tokyo, 152-8551, Japan
satohcgn@mathpc-satoh.math.titech.ac.jp

Abstract. In hyperelliptic curve cryptography, finding a suitable hyperelliptic curve is an important fundamental problem. One of necessary conditions is that the order of its Jacobian is a product of a large prime number and a small number. In the paper, we give a probabilistic polynomial time algorithm to test whether the Jacobian of the given hyperelliptic curve of the form $Y^2 = X^5 + uX^3 + vX$ satisfies the condition and, if so, to give the largest prime factor. Our algorithm enables us to generate random curves of the form until the order of its Jacobian is almost prime in the above sense. A key idea is to obtain candidates of its zeta function over the base field from its zeta function over the extension field where the Jacobian splits.

Keywords: hyperelliptic curve, point counting.

1 Introduction

In (hyper)elliptic curve cryptography, point counting algorithms are very important to exclude the weak curves. For elliptic curves over finite fields, the SEA algorithm (see Schoof[36] and Elkies[8]) runs in polynomial time (with respect to input size). In case that the characteristic of the coefficient field is small, there are even faster algorithms based on the p-adic method (see e.g. Vercauteren[40] for a comprehensive survey). The p-adic method gives quite efficient point counting algorithms for higher dimensional objects, e.g. Kedlaya[22], Lercier and Lubicz[26], Lauder[23]. However, so far, there is no known efficient practical algorithm for hyperelliptic curves of genus two in case that the characteristic of the coefficient field is large.

In theory, Pila[32] generalized the Schoof algorithm to a point counting algorithm for Abelian varieties over finite fields. Nevertheless, a pure polynomial time algorithm has not been successfully used even for the Jacobian of hyperelliptic curves of genus two. Current implementations for crypto size curves of genus two more or less contain the BSGS process, hence the running time grows exponentially. For details on implementation, see Matsuo, Chao and Tsujii[27], Gaudry and Schost[17,18] and references cited there.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* The work was supported by the Grant-in-Aid for the Scientific Research (B) 18340005.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 536–553, 2009.

On the other hand, using basic properties on character sums (see e.g. Berndt, Evans and Williams[3] for standard facts on the topic), Furukawa, Kawazoe and Takahashi[11] gives an explicit formula for the order of Jacobians of curves of type $Y^2 = X^5 + aX$ where $a \in \mathbf{F}_p^\times$. However, there are, at most, only 8 isomorphism classes over $\mathbf{F}_p$ among these curves for each prime p. In order to obtain a curve suitable for cryptography, they try various values of p until a suitable curve is found. Their method relies on the binomial expansion of $X^5 + aX$. The idea was generalized to hyperelliptic curves over prime fields of the form $Y^2 = X^5 + a$ (ibid.) and $Y^2 = X^{2k+1} + aX$ in Haneda, Kawazoe and Takahashi[19]. Recently, Anuradha[2] obtained similar formulae for non-prime fields. But their method seems to be applicable only to binomials in X.

In this paper, we consider an intermediate case: our curve is in a certain special form but not as special as was considered in the above papers and time complexity to test one curve is of probabilistic polynomial time. More specifically, we give an algorithm to test whether the order of the Jacobian of a given hyperelliptic curve of genus two in the form $Y^2 = X^5 + uX^3 + vX$ has a large prime factor. If so, the algorithm outputs the prime factor. Moreover, under a certain condition (see Remark 5) which is always satisfied in cryptographic applications, our algorithm determines the group order itself. Numerical experiments suggest that curves in this form which are suitable for cryptography exist for any large p. We may attempt to choose the base field for which arithmetic operations can be efficiently performed.

The order of the Jacobian of hyperelliptic curves $Y^2 = X(X^{2n} + uX^n + v)$ over a prime field are already studied by Leprévost and Morain[24]. In case of $n = 2$, they gave some explicit formulae for the order of the Jacobian in terms of certain modular functions, whose evaluations are computationally feasible only for special combinations of u and v.

Our method is totally different from the preceding point counting works. Let $p \geq 5$ be a prime and let q be a power of p. However, it is recommended to take $q = p$ to avoid a possible attack due to Gaudry[15] (cf. Section 8). Let $C/\mathbf{F}_q : Y^2 = X^5 + uX^3 + vX$ be an arbitrary (in view of cryptographic application, randomly) given hyperelliptic curve. We now observe a key idea of our method. Put $r = q^4$. We denote the Jacobian variety of C by J, which is an Abelian variety of dimension two defined over $\mathbf{F}_q$. Now J is $\mathbf{F}_r$-isogenous to a square of an elliptic curve defined over $\mathbf{F}_r$. Hence the (one dimensional part of the) zeta function of J as an Abelian variety over $\mathbf{F}_r$ is a square of the zeta function of the elliptic curve, which is computed by the SEA algorithm. On the other hand, we have some relation between the zeta function of J over $\mathbf{F}_r$ and over $\mathbf{F}_q$. This gives (at most 26) possible orders of $J(\mathbf{F}_q)$. For each candidate, we first check that the order is not weak for cryptographic use, that is, the order is a product of a small positive integer and a large prime. If the curve is weak, we stop here. (Note that, if the curve passes the test, its Jacobian is simple over $\mathbf{F}_q$.) Then, we take random points of $J(\mathbf{F}_q)$ to see whether the prime actually divides $\#J(\mathbf{F}_q)$. Our algorithm runs in probabilistic polynomial time in $\log q$. In order

to find a hyperelliptic curve suitable for cryptography, we repeat the process with randomly given u and v until we obtain a curve with desired properties.

In the case of characteristics two, Hess, Seroussi and Smart[20] proposed an algorithm using Weil descent to construct verifiably randomly a hyperelliptic curve in a certain family which is suitable for a hyperelliptic cryptosystem. The efficiency of their algorithm in case of large characteristics is not clear. As to products of elliptic curves, Scholten[35] randomly constructs a hyperelliptic curve of genus two for odd prime fields whose Jacobian is isogenous to the Weil restriction of elliptic curves. Both of the works start with elliptic curves while our algorithm starts with the hyperelliptic curve $Y^2 = X^5 + uX^3 + vX$ where u and v are given.

Recently Sutherland[38] proposed an algorithm based on a generic group model to produce Abelian varieties from random hyperelliptic curves. When applied to curves of genus two, its running time is quite practical but its heuristic time complexity is of sub-exponential. Although these two algorithms [20,38] take random input data, it is highly non-trivial to observe distribution of output of the algorithm. In case of our algorithm, it is obvious that our algorithm generates each curve of type $Y^2 = X^5 + uX^3 + vX$, suitable to cryptography with equal probability if we generate random u and v uniformly.

The Jacobian has complex multiplications by $\sqrt{-1}$ for all u and v as long as the curve is actually a hyperelliptic curve. This fact can be used to make scalar multiplication faster by the Gallant, Lambert and Vanstone algorithm[12]. We can also take advantage of real multiplications with efficient evaluations (if any) due to Takashima[39]. However we also note that such an efficient endomorphism also speeds up solving discrete log problems Duursma, Gaudry and Morain[7].

In case of $q \equiv 1 \bmod 4$, we can apply our algorithm to $Y^2 = X(X^2-\alpha^2)(X^2-\beta^2)$ for randomly generated α, $\beta \in \mathbf{F}_q^\times$. (When $q \equiv 3 \bmod 4$, such curves are never suitable for cryptography. See Remark 2.) All 2-torsion points of the Jacobian of such a curve are $\mathbf{F}_q$-rational. Hence, the efficient scalar multiplication algorithm due to Gaudry[14] is applicable to them. Although it is not clear how often such a curve is suitable for cryptography, a numerical experiment found a few suitable curves.

After the initial submission for the conference, the author learned that Gaudry and Schost[16] completely describes the hyperelliptic curves whose Jacobian is isogenous to a product of two elliptic curves by an isogeny with kernel $\mathbf{Z}/2\mathbf{Z} \oplus \mathbf{Z}/2\mathbf{Z}$. Our idea is probably applicable to such curves given by in terms of the Rosenhain form or the invariants (Ω, Υ) in the notation of [16]. The j-invariants of two elliptic curves will be different but no essential change to our idea is required. However, the author have not derived explicit formulae for these parameters, yet. Recently, Paulhus[31] gave explicit descriptions of splitting of Jacobian into elliptic curves for some curves of genus greater than two. Although the use of curves with genues greater than four in cryptography is rather questionable in view of the index calculus method due to Gaudry[13], it would be interesting to consider number theoretic properties of the order of such curves.

The rest of the paper is organized as follows. In Section 2, we review some facts on arithmetic properties of the Jacobian varieties. In Section 3, we give an explicit formula of the elliptic curve whose power is isogenous to the Jacobian of the given hyperelliptic curve of the above form. In Section 4, we show how to retrieve possible orders of the Jacobian via the decomposition obtained in the preceding section. In Section 5, we state our algorithm and observe its computational complexity. In Section 6, we give an illustrative small numerical example. In Section 7, we report results of numerical experiments with cryptographic size parameters. We observe some security implication of our curves in section 8.

Throughout the paper, we let p be a prime greater than or equal to 5 and q a power of p. We put $r = q^4$.

2 Some Properties of the Jacobian Varieties

We summarize arithmetic properties of the Jacobian varieties used in the later sections. See the surveys Milne[29,30] for more descriptions.

Let $A/\mathbf{F}_q$ be an Abelian variety of dimension d. The (one dimensional part of the) zeta function $Z_A(T, \mathbf{F}_q)$ is the characteristic polynomial of the q-th power Frobenius map on $V_l(A) = T_l(A) \otimes_{\mathbf{Z}_l} \mathbf{Q}_l$ where l is a prime different from p and $T_l(A)$ is the l-adic Tate module. It is known that $Z_A(T, \mathbf{F}_q) \in \mathbf{Z}[T]$ with $\deg Z_A(T, \mathbf{F}_q) = 2d$ and that it is independent of the choice of l. It holds that

$$\#A(\mathbf{F}_q) = Z_A(1, \mathbf{F}_q). \tag{1}$$

Let $\prod_{i=1}^{2d}(T - z_{i,q})$ be the factorization of $Z_A(T, \mathbf{F}_q)$ in $\mathbf{C}[T]$. Permuting indices if necessary, we may assume that

$$z_{1,q}z_{2,q} = q, \quad \ldots, \quad z_{2d-1,q}z_{2d,q} = q. \tag{2}$$

Let $n \in \mathbf{N}$ and put $\tilde{q} = q^n$. Since the $\tilde{q}$-th power map is the n-times iteration of the q-th power map, we see

$$\{z_{1,\tilde{q}}, z_{2,\tilde{q}}, \ldots, z_{2d,\tilde{q}}\} = \{z_{1,q}^n, z_{2,q}^n, \ldots, z_{2d,q}^n\} \tag{3}$$

including multiplicity. It holds that $|z_{i,\tilde{q}}| = \sqrt{\tilde{q}}$.

In case that A is isogenous to $A_1 \times A_2$ over $\mathbf{F}_q$, we have $V_l(A) \cong V_l(A_1) \oplus V_l(A_2)$ as $\mathrm{Gal}(\overline{\mathbf{F}_q}/\mathbf{F}_q)$-modules. Hence

$$Z_A(T, \mathbf{F}_q) = Z_{A_1}(T, \mathbf{F}_q) Z_{A_2}(T, \mathbf{F}_q). \tag{4}$$

Let $E/\mathbf{F}_q$ be an elliptic curve, which is an Abelian variety of dimension 1 over $\mathbf{F}_q$. The above items translate to the well known formula

$$Z_E(T, \mathbf{F}_q) = T^2 - tT + q$$

with $|t| \leq 2\sqrt{q}$ where $t = q + 1 - \#E(\mathbf{F}_q)$.

Let $C/\mathbf{F}_q$ be a hyperelliptic curve of genus two and let J be its Jacobian variety, which is an Abelian variety of dimension two defined over $\mathbf{F}_q$. Let $z_{1,q}$, ..., $z_{4,q}$ be the roots of $Z_J(T, \mathbf{F}_q)$ arranged as in (2). We see

$$Z_J(T, \mathbf{F}_q) = T^4 - a_q T^3 + b_q T^2 - q a_q T + q^2 \tag{5}$$

where

$$a_q = \sum_{i=1}^{4} z_{i,q}, \qquad b_q = \sum_{i=1}^{3} \sum_{j=i+1}^{4} z_{i,q} z_{j,q}.$$

We note $|a_q| \leq 4\sqrt{q}$ and $|b_q| \leq 6q$. We will also use the Hasse-Weil bound

$$(\sqrt{q} - 1)^4 \leq \#J_C(\mathbf{F}_q) \leq (1 + \sqrt{q})^4. \tag{6}$$

3 Decomposition of the Jacobian

In this section, we give an explicit formula of two elliptic curves whose product is isogenous to the Jacobian of the hyperelliptic curves of our object. Such a decomposition has been more or less known, e.g. Leprévost and Morain[24], Cassel and Flynn[5, Chap. 14], and Frey and Kani[9]. Here we derive a formula which is ready to implement our method efficiently.

Let $C : Y^2 = X^5 + uX^3 + vX$ be a hyperelliptic curve where $u \in \mathbf{F}_q$ and $v \in \mathbf{F}_q^\times$. We denote the Jacobian variety of C by J. There exist α, $\beta \in \mathbf{F}_r^\times$ such that

$$X^5 + uX^3 + vX = X(X^2 - \alpha^2)(X^2 - \beta^2).$$

We choose and fix $s \in \mathbf{F}_{q^8}^\times$ satisfying $s^2 = \alpha\beta$. In fact, $s \in \mathbf{F}_r^\times$ since $s^4 = \alpha^2\beta^2 = v \in \mathbf{F}_q^\times$. It is straightforward to verify

$$\begin{aligned} X^2 + (\alpha + \beta)X + \alpha\beta &= A(X + s)^2 + B(X - s)^2 \\ X^2 - (\alpha + \beta)X + \alpha\beta &= B(X + s)^2 + A(X - s)^2 \end{aligned}$$

where

$$\begin{aligned} A &= \frac{1}{2}\left(1 + \frac{\alpha + \beta}{2s}\right), \\ B &= \frac{1}{2}\left(1 - \frac{\alpha + \beta}{2s}\right). \end{aligned}$$

Then

$$\begin{aligned} X^4 + uX^2 + v &= (X^2 - \alpha^2)(X^2 - \beta^2) = (X + \alpha)(X + \beta)(X - \alpha)(X - \beta) \\ &= AB\left((X + s)^4 + \left(\frac{B}{A} + \frac{A}{B}\right)(X + s)^2(X - s)^2 + (X - s)^4\right). \end{aligned}$$

Note that X can be rewritten in terms of $X - s$ and $X + s$:

$$X = \frac{1}{4s}\left((X + s)^2 - (X - s)^2\right).$$

Define $E_1/\mathbf{F}_r$ and $E_2/\mathbf{F}_r$ by

$$E_1 : Y^2 = \delta(X-1)(X^2-\gamma X+1)$$
$$E_2 : Y^2 = -\delta(X-1)(X^2-\gamma X+1)$$

where

$$\delta = \frac{AB}{4s} = -\frac{(\alpha-\beta)^2}{64s^3}, \tag{7}$$
$$\gamma = -\left(\frac{B}{A}+\frac{A}{B}\right) = 2(\alpha^2+6\alpha\beta+\beta^2)/(\alpha-\beta)^2. \tag{8}$$

Then, we have two covering maps $\varphi_i : C \to E_i$ defined over $\mathbf{F}_r$ by

$$\varphi_1(x,y) = \left(\left(\frac{x+s}{x-s}\right)^2, \frac{y}{(x-s)^3}\right),$$
$$\varphi_2(x,y) = \left(\left(\frac{x-s}{x+s}\right)^2, \frac{y}{(x+s)^3}\right).$$

They induce maps $\varphi_i^* : \mathrm{Div}(E_i) \to \mathrm{Div}(C)$ and $\varphi_{i*} : \mathrm{Div}(C) \to \mathrm{Div}(E_i)$. They again induce maps (which are also denoted by) $\varphi_i^* : \mathrm{Pic}^0(E_i)\ (\cong E) \to \mathrm{Pic}^0(C)\ (\cong J)$ and $\varphi_{i*} : J \to E_i$. We note that $\varphi_{i*} \circ \varphi_i^*$ is the multiplication by 2 map on E_i and that both $\varphi_{1*} \circ \varphi_2^*$ and $\varphi_{2*} \circ \varphi_1^*$ are the zero maps. Therefore J is isogenous to $E_1 \times E_2$.

Since $2|[\mathbf{F}_r : \mathbf{F}_p]$ and $p \geq 5$, both E_1 and E_2 are isomorphic to the following elliptic curve in the short Weierstrass from:

$$E : Y^2 = X^3 - \frac{(\gamma-2)(\gamma+1)}{3}\delta^2 X - \frac{(\gamma-2)^2(2\gamma+5)}{27}\delta^3. \tag{9}$$

Eventually, J is isogenous to $E \times E$ over $\mathbf{F}_r$. Using (4), we conclude

$$Z_J(T, \mathbf{F}_r) = Z_E(T, \mathbf{F}_r)^2. \tag{10}$$

Remark 1. Observe that in fact $\gamma \in \mathbf{F}_{q^2}$. By (8), we see that $\gamma \in \mathbf{F}_q$ if and only if either $u = 0$ or $u \neq 0$ and $\alpha\beta \in \mathbf{F}_q^\times$ (i.e. v is a square element in $\mathbf{F}_q$). Thus, we do not need to run the SEA algorithm to $E/\mathbf{F}_r$. Define $E'/\mathbf{F}_q(\gamma)$ by

$$E' : Y^2 = X^3 - \frac{(\gamma-2)(\gamma+1)}{3}X - \frac{(\gamma-2)^2(2\gamma+5)}{27}. \tag{11}$$

Let σ be the trace of the $\#\mathbf{F}_q(\gamma)$-th power Frobenius endomorphism on E'. We obtain σ by running the SEA algorithm to $E'/\mathbf{F}_q(\gamma)$. Note that the size of the coefficient field is smaller than the size of $\mathbf{F}_r$ by a factor of $1/2$ or $1/4$. Let τ' be the trace of the r-th power Frobenius endomorphism on E'. Then

$$\tau' = \begin{cases} (\sigma^2-2q)^2 - 2q^2 & (\gamma \in \mathbf{F}_q), \\ \sigma^2 - 2q^2 & (\gamma \notin \mathbf{F}_q). \end{cases}$$

Now E is isomorphic to E' over $\mathbf{F}_{r^2}$. Let τ be the trace of the r-th power Frobenius endomorphism on E. Unless $j(E) = 0$ or $j(E) = 1728$, we have

$$\tau = \begin{cases} \tau' & (\delta^{(r-1)/2} = 1), \\ -\tau' & (\delta^{(r-1)/2} = -1). \end{cases}$$

(And it is easy to compute τ in case of $j(E) = 0$ or $j(E) = 1728$, see e.g. Schoof[37, Sect. 4].) This device does not affect growth rate of the computational complexity of our algorithm. However practical performance improvement by this is significant, since the most of computational time is spent for the SEA algorithm.

Remark 2. Note that, in fact, E_1 and E_2 are defined over $\mathbf{F}_q(s)$. Changing the sign of α if necessary in case of $q \equiv 3 \bmod 4$, we see $s \in \mathbf{F}_q$ when v is a fourth power element of $\mathbf{F}_q^\times$. In this case J already splits over $\mathbf{F}_q$. Note that in case of $q \equiv 3 \bmod 4$, any square element in $\mathbf{F}_q^\times$ is a fourth power element.

4 Computing Possible Order of the Jacobian

In this section, we consider how to obtain $Z_J(T, \mathbf{F}_q)$ from $Z_J(T, \mathbf{F}_r)$. Actually, this is quite elementary.

Let $z_{1,q}, \ldots, z_{4,q}$ be the roots of $Z_J(T, \mathbf{F}_q)$ arranged as (2). Put $s_n = \sum_{i=1}^4 z_{i,q}^n$ with a convention $s_0 = 4$. Then

$$\begin{aligned} s_1 &= a_q, \\ s_2 &= a_q^2 - 2b_q, \\ s_3 &= a_q^3 - 3a_q b_q + 3q a_q \end{aligned}$$

and

$$s_i = a_q s_{i-1} - b_q s_{i-2} + q a_q s_{i-3} - q^2 s_{i-4}$$

for $i \geq 4$. In particular, we obtain

$$\begin{aligned} s_4 &= a_q^4 - 4(b_q - q)a_q^2 + 2b_q^2 - 4q^2, \\ s_8 &= a_q^8 - 8(b_q - q)a_q^6 + (20b_q^2 - 32qb_q + 4q^2)a_q^4 \\ &\quad + (-16b_q^3 + 24qb_q^2 + 16q^2 b_q - 16q^3)a_q^2 + 2b_q^4 - 8q^2 b_q^2 + 4q^4. \end{aligned}$$

Recall that $r = q^4$. Hence $a_r = s_4$ and

$$b_r = (s_4^2 - s_8)/2 = 2q^2 a_q^4 + (-4qb_q^2 + 8q^2 b_q - 8q^3)a_q^2 + b_q^4 - 4q^2 b_q^2 + 6q^4.$$

Recall that J is isogenous to $E \times E$ over $\mathbf{F}_r$ where E is defined by (11). Let t be the trace of r-th Frobenius map on E. Then, $Z_E(T, \mathbf{F}_r) = T^2 - tT + r$. Thus (10) gives

$$T^4 - a_r T^3 + b_r T^2 - \cdots = T^4 - 2tT^3 + (2r + t^2)T^2 + \cdots,$$

that is

$$\begin{aligned} 0 = {} & a_q^4 - 4(b_q - q)a_q^2 + 2b_q^2 - 4q^2 - 2t, \qquad (12)\\ 0 = {} & 2q^2a_q^4 + (-4qb_q^2 + 8q^2b_q - 8q^3)a_q^2 + b_q^4 - 4q^2b_q^2 + 6q^4 - (2q^4 + t^2). \end{aligned}$$

Eliminating b_q by computing a resultant of the above two polynomials, we obtain

$$\begin{aligned} a_q^{16} - {} & 32qa_q^{14} + (368q^2 - 8t)a_q^{12} + (-1920q^3 + 64tq)a_q^{10} \\ & + (4672q^4 + 64tq^2 - 112t^2)a_q^8 + (-5120q^5 - 1024tq^3 + 768t^2q)a_q^6 \\ & + (2048q^6 + 1024tq^4 - 512t^2q^2 - 256t^3)a_q^4 = 0. \qquad (13) \end{aligned}$$

This yields at most 13 possible values for a_q. Note that a_q is an integer satisfying $|a_q| \leq 4\sqrt{q}$. In order to find integer solutions of a_q, we choose any prime l satisfying $l > 8\sqrt{q}$ and factor the above polynomial in $\mathbf{F}_l$. We only need linear factors. For each $\mathbf{F}_l$-root, we check whether it is a genuine root in characteristic zero and its absolute value does not exceed $4\sqrt{q}$. For each possible a_q, we easily obtain at most two possible values of b_q satisfying the above equations. Thus, we have obtained at most 26 candidates for $\#J(\mathbf{F}_q)$.

5 The Algorithm and Its Complexity

In this section, we analyze a time computational complexity of our algorithm. First, we state our method in a pseudo-code. Then, we show our algorithm terminates in probabilistic polynomial time in $\log q$. We denote the identity element of J by 0 in the following.

In addition to the coefficients of hyperelliptic curve C, we give two more inputs: the "cofactor bound" M and a set of "test points" D, which is any subset of $J(\mathbf{F}_q)$ satisfying $\#D > M$.

In order that the discrete logarithm problem on $J(\mathbf{F}_q)$ is not vulnerable to the Pohlig-Hellman attack[33], $\#J(\mathbf{F}_q)$ must be divisible by a large prime (at least 160 bit in practice). We request that the largest prime factor is greater than $\#J(\mathbf{F}_q)/M$, which ensures that the factor is greater than $(\sqrt{q}-1)^4/M$. In the following algorithm, M must be less than $(\sqrt{q}-1)^2$. In practice, $M < 2^8$ (at most) in view of efficiency of group operation. So, building up D is easy for such small M.

Algorithm 1
Input: Coefficients u, $v \in \mathbf{F}_q$ in $C : Y^2 = X^5 + uX^3 + vX$,
a cofactor bound $M \in \mathbf{N}$ satisfying $M < (\sqrt{q}-1)^2$,
a subset D of $J(\mathbf{F}_q)$ satisfying $\#D > M$.
Output: The largest prime factor of $\#J(\mathbf{F}_q)$ if it is greater than $\#J(\mathbf{F}_q)/M$. Otherwise, **False**.
Procedure:
1: Let α_0, $\beta_0 \in \mathbf{F}_{q^2}$ be the solution of $x^2 + ux + v = 0$.
2: Find $\alpha \in \mathbf{F}_r$ and $\beta \in \mathbf{F}_r$ satisfying $\alpha^2 = \alpha_0$, $\beta^2 = \beta_0$.

3: Compute δ and γ by (7) and (8), respectively.
4: Compute $\#E(\mathbf{F}_r)$ by the SEA algorithm and put $t = 1 + q^r - \#E(\mathbf{F}_r)$.
5: Find a prime l satisfying $8\sqrt{q} < l \leq q$.
6: Find solutions of (13) modulo l.
7: for each solution τ do:
8: Lift $\tau \in \mathbf{F}_l$ to $a_q \in \mathbf{Z}$ so that $|a_q| \leq 4\sqrt{q}$.
9: if a_q is really a root of (13) and if a_q is even then
10: for each integer solution b_q of (12) satisfying $|b_q| \leq 6q$ do:
11: $L = 1 - a_q + b_q - qa_q + q^2$ /* cf. (1), (5) */
12: if $(\sqrt{q}-1)^4 \leq L \leq (\sqrt{q}+1)^4$ then /* cf. (6) */
13: Find the largest divisor d of L less than M.
14: $L' = L/d$
15: if (L' is prime) then
16: Find a point $P \in D$ such that $dP \neq 0$.
17: if $LP = 0$, then output L' and stop.
18: endif
19: endif /* L satisfies the Hasse-Weil bound */
20: endfor /* b_q */
21: endif
22: endfor /* τ */
23: Output **False** and stop.

Remark 3. Actually, in the SEA algorithm in Step 4, we obtain t before we obtain $\#E(\mathbf{F}_q)$.

Remark 4. Instead of giving a set D by listing all points, we can specify D by some conditions and we generate elements of D during execution of the above procedure. In the implementation used in the next section, the author used

$$D = \{[P] + [Q] - 2[\infty] \ : \ P,\ Q \in C(\mathbf{F}_q) - \{\infty\},\ P_X \neq Q_X\}$$

where ∞ is the point at infinity of C (not J) and the subscript X stands for the X-coordinate. Then, $\#D \approx O(q^2)$. It is easy to generate a uniformly random point of D in probabilistic polynomial time.

Remark 5. In case that $d \leq \frac{\sqrt{q}}{8} - \frac{1}{2}$ (which always holds when $M \leq \frac{\sqrt{q}}{8} - \frac{1}{2}$), the value of L at Step 17 gives $\#J(\mathbf{F}_q)$. The reason is as follows. Observe that

$$\left(\frac{x}{8} - \frac{1}{2}\right)((x+1)^4 - (x-1)^4) < (x-1)^4$$

for $x \in \mathbf{R}$. Thus

$$L' \geq \frac{(\sqrt{q}-1)^4}{d} \geq \frac{(\sqrt{q}-1)^4}{\frac{\sqrt{q}}{8} - \frac{1}{2}} > (\sqrt{q}+1)^4 - (\sqrt{q}-1)^4.$$

Hence, there is only one multiple of L' in the Hasse-Weil bound (6), which must be $\#J(\mathbf{F}_q)$.

Remark 6. Instead of Steps 5–6, one can choose a small prime l which does not divide the discriminant of the square free part of (13), and then factor (13) modulo l and lift the factors to modulo l^n where $l^n > 8\sqrt{q}$. See e.g. von zur Gathen and Gerhard[42] for details of both methods and comparison. In theory, both methods run in polynomial time w.r.t. input size. In practice, time for these steps are negligible for crypto size parameters. The use of a large prime is easy to implement and conceptually simple. One theoretical concern is the number of primality tests to find l. As we will see (14) below, there exist at least $\Omega(q/\log q)$ primes l satisfying $8\sqrt{q} < l < q$. Thus, average number of primality tests to find l in Step 5 is $O(\log q)$. In an actual implementation, we may search for a prime by testing odd numbers greater than $8\sqrt{q}$ one by one. Primality tests can be performed by a deterministic polynomial time algorithm by Agrawal, Kayal and Saxena[1]. Since our main interest is in hyperelliptic cryptography, we do not go into complexity arguments on primality tests further.

Theorem 1. *Let $M < (\sqrt{q}-1)^2$ be fixed. Then Algorithm 1 terminates in probabilistic polynomial time in* $\log q$ *(with respect to the bit operations). In case that $\#J(\mathbf{F}_q)$ is divisible by a prime greater than $(\sqrt{q}-1)^4/M$, Algorithm 1 returns the largest prime factor of $\#J(\mathbf{F}_q)$.*

Proof. Note that t, q, a_q and b_q are integers. Hence a_q must be even by (12). This explains Step 9. If we reach Step 16, we have, as an Abelian group,

$$J(\mathbf{F}_q) \cong G \oplus (\mathbf{Z}/L'\mathbf{Z})$$

where G is an Abelian group of order d. Since $\gcd(d, L') = 1$, there are at most d points $Q \in J(\mathbf{F}_q)$ satisfying $dQ = 0$. Since $\#D > M$, we can find P at Step 16 by testing at most M elements in D. Note $\#J(\mathbf{F}_q) \geq (\sqrt{q}-1)^4$. Therefore,

$$L' \geq \frac{\#J(\mathbf{F}_q)}{M} \geq \frac{\#J(\mathbf{F}_q)}{(\sqrt{q}-1)^2} \geq \sqrt{\#J(\mathbf{F}_q)}.$$

Since L' is a prime, it must be the largest prime factor of $\#J(\mathbf{F}_q)$, which completes the proof of correctness of the algorithm.

Now we consider computational complexity. Note that a bit size of any variable appearing in Algorithm 1 is bounded by $c \log q$ where c is a constant depending only on M and the primality test algorithm used in Step 5. Thus we have only to show that, instead of the number of bit operations, the number of arithmetic operations performed in Algorithm 1 is bounded by a polynomial in $\log q$. For a positive real number x, put $\theta(x) = \sum_{l \leq x} \theta(x)$ as usual where l runs over primes less than x. By Chebyshev's theorem[6], there exist constants C_1 and C_2 such that

$$Kx - C_1\sqrt{x}\log x < \theta(x) < \frac{6}{5}Kx$$

for all $x > C_2$ where $K = \log \frac{2^{1/2}3^{1/3}5^{1/5}}{30^{1/30}}$ $(= 0.921...)$. Let $\nu(q)$ be the number of primes l satisfying $8\sqrt{q} < l \leq q$. Then

$$\nu(q)\log q \geq \sum_{8\sqrt{q} < l \leq q} \log l = \theta(q) - \theta(8\sqrt{q})$$

and thus

$$\nu(q) \geq K\frac{q}{\log q} - C_3\sqrt{q} \text{ for } q > C_4 \tag{14}$$

with some C_3, $C_4 > 0$. Therefore, if we test random odd numbers between $8\sqrt{q}$ and q to find l, an average number of primality tests in Step 5 is less than $\frac{\log q}{2K}\left(1 + \frac{2C_3 \log q}{K\sqrt{q}}\right)$ for all $q > C_4$. In Steps 1, 2 and 6, we need to factor univariate polynomials over $\mathbf{F}_r$ or $\mathbf{F}_l$. However, the degree of polynomials to be factored is either 2 or 13. Hence the Cantor-Zassenhaus factorization[4] factors them in probabilistic polynomial time. The SEA algorithm (even Schoof's algorithm alone) runs in polynomial time. Summing up, we obtain our assertions. □

6 A Numerical Example

We give an illustrative example of our algorithm. We set $M = 16$. Let $q = p = 509$ and consider $C : Y^2 = X^5 + 3X^3 + 7X$. Then, $\mathbf{F}_r = \mathbf{F}_p(\theta)$ where $\theta^4 + 2 = 0$. For simplicity, we write an element $\mu_3\theta^3 + \mu_2\theta^2 + \mu_1\theta + \mu_0$ of $\mathbf{F}_r$ as $[\mu_3\ \mu_2\ \mu_1\ \mu_0]$. Then we have $\alpha = [193\ 0\ 90\ 0]$, $\beta = [67\ 0\ 396\ 0]$, $s = [0\ 0\ 427\ 0]$, $\delta = [29\ 0\ 488\ 0]$ and $\gamma = [0\ 56\ 0\ 17]$. Hence the short Weierstrass form of E is

$$Y^2 = X^3 + [0\ \ 370\ \ 0\ \ 73]X + [293\ \ 0\ \ 464\ \ 0].$$

An elliptic curve point counting algorithm gives $t = 126286$. We take $l = 191$. Then, (13) in this example is

$$a_q^{16} - 16288a_q^{14} + 94331520a_q^{12} - 249080786944a_q^{10} + 313906268606464a_q^8 \\ -185746793889398784a_q^6 + 41664329022490804224a_q^4 = 0.$$

Reducing modulo l, we obtain

$$\begin{aligned} 0 &= \overline{a_q}^{16} + 138\overline{a_q}^{14} + 58\overline{a_q}^{12} + 46\overline{a_q}^{10} + 63\overline{a_q}^8 + 94\overline{a_q}^6 + 136\overline{a_q}^4 \\ &= (\overline{a_q}^2 + 56\overline{a_q} + 170)(\overline{a_q}^2 + 135\overline{a_q} + 170)(\overline{a_q}^2 + 150)^2(\overline{a_q} + 28)^2(\overline{a_q} + 163)^2\overline{a_q}^4 \end{aligned}$$

where $\overline{a_q}$ is the reduction of a_q modulo l. Hence $a_q = -28$, 0, 28. In case of $a_q = -28$, Eq. (12) is $b_q^2 - 784b_q + 460992 = 0$. Thus $b_q = 1176$, 392. The former values gives $L = 274538 = 11 \cdot 24958$ and 24958 is apparently not prime. Similarly, the case $b_q = 392$ gives $L = 273754 = 13 \cdot 21058$. In case of $a_q = 0$, Eq. (12) does not have an integer solution. Finally, we consider the case $a_q = 28$. The equation (and hence the solution) for b_q is the same as that for $a_q = -28$. Now $b_q = 1176$ gives $L = 245978 = 2 \cdot 122989$ and $122989 = 29 \cdot 4241$ is not prime. But in the case of $b_q = 392$, we obtain $L = 245194 = 2 \cdot 122597$ and 122597 is prime. Take $P = (X^2 + 286X + 46, 347X + 164) \in J(\mathbf{F}_p)$ written in the Mumford representation. Then, $2P = (X^2 + 365X + 23, 226X + 240) \neq 0$ but $LP = 0$. Thus, 122597 is the largest prime divisor of $\#J(\mathbf{F}_p)$. In this example, we also conclude $\#J(\mathbf{F}_p) = 245194$ because $d = 2 \leq \frac{\sqrt{509}}{8} - \frac{1}{2} = 2.32...$

7 Cryptographic Size Implementation

In this section, we report implementation results for cryptographic size parameters. The results show that our algorithm certainly produces hyperelliptic curves for cryptographic use with an acceptable computational complexity. In reality, almost all computation time is consumed in an elliptic curve point counting (Step 4 in Algorithm 1). We begin with remarks on an elliptic curve point counting.

First, deployment of Remark 1 is very important. Assume that we use the Karatsuba algorithm for multiplications. Then Remark 1 reduces running time by a factor of $2^{2+2\log_2 3} \approx 36.2$ in theory. More importantly, a number of modular polynomials necessary for the SEA algorithm is approximately halved.

Next the action of the Frobenius map can be computed more efficiently than a straightforward method. Let p be a prime and let n be an integer greater than 1. In this paragraph, we put $q = p^n$. Assume that we use a variant of the baby-step giant-step algorithm due to Maurer and Müller[28] to find an eigenvalue of the Frobenius endomorphism. Then a bottleneck of Elkies' point counting algorithm for elliptic curves defined over $\mathbf{F}_q$ is a computing action of the q-th power map and $\left(\frac{q-1}{2}\right)$-th power map in $\mathbf{F}_q[t]/\langle b(t)\rangle$ for some $b(t) \in \mathbf{F}_q[t]$. To reduce their computational time, we use an algorithm due to von zur Gathen and Shoup[43] with a modification to take the action of p-th power map on $\mathbf{F}_q$ into consideration. See Vercauteren[41, Sect. 3.2] in the case $p = 2$. Even in the case of $n = 2$ (which is the case we need), the modification saves much time. The overall performance improvement of elliptic curve point counting with this technique highly depends on an implementation and field parameters. In author's implementation (which uses the Elkies primes only) for $p \approx 2^{87}$ and $n = 2$, the improvement was approximately in range $20 \sim 40\%$ depending on elliptic curves.

Now we back to our algorithm. In what follows, q is the cardinality of the coefficient field of hyperelliptic curves to be generated. Assume that we need the Jacobian variety whose bit size of the group order is N. Then, $\log q \cong N/2$. Note we apply the SEA algorithm to elliptic curves over $\mathbf{F}_{q^2}$. Thus we can paraphrase that time to test one random hyperelliptic curve of the form $Y^2 = X^5 + uX^3 + vX$ is comparable to time to test one random elliptic curve if their group sizes are the same. Of course, this is only rough comparison. For example, in the elliptic curve case, we can employ early abort strategy (see Lercier[25, Sect. 4.1]), whereas we need to complete the SEA algorithm in our Algorithm 1.

Since our curves $Y^2 = X^5 + uX^3 + vX$ are in rather special form, they may be less likely suitable for cryptographic use. The author has no idea for theoretical results on this point. To observe the circumstance, the following numerical experiments are performed. Take a 87 bit number $A =$ 0x5072696d654e756d626572 in hexadecimal (the ASCII code for the string "PrimeNumber"). For the largest five primes p less than A satisfying $p \equiv 1 \bmod 4$ and $p \equiv 3 \bmod 4$, that is, for each prime $A - 413$, $A - 449$, $A - 609$, $A - 677$ and $A - 957$ (they are 1 mod 4) and $A - 23$, $A - 35$, $A - 39$, $A - 179$, $A - 267$ (they are 3 mod 4), Algorithm 1 is executed for 200 randomly generated $(u, v) \in \mathbf{F}_q \times (\mathbf{F}_q^\times - \mathbf{F}_q^4)$ (cf. Remark 2). In Table 1, we list the cofactors obtained in the numerical experiments. For each

Table 1. Distributions of cofactor returned by the algorithm

p	cofactors
$A-413$	2, 4, 4, 10, 10, 16, 16, 20, 40 80, 130, 208, 400, 580
$A-449$	2, 4, 4, 4, 26, 120, 394, 722, 740
$A-609$	2, 20, 52, 116, 256, 484, 820
$A-677$	4, 4, 8, 10, 20, 26, 34, 40, 82, 362, 400, 482, 740
$A-957$	2, 4, 4, 16, 20, 20, 20, 72, 90, 100, 262, 720
$A-23$	2, 2, 2, 14, 14, 16, 16, 34, 112, 184, 302
$A-35$	2, 8, 14, 18, 34, 56, 72, 194, 392
$A-39$	2, 2, 2, 8, 16, 62, 94, 98, 584, 656, 752, 784
$A-179$	2, 8, 18, 18, 32, 128, 146, 386, 512
$A-267$	2, 8, 8, 8, 14, 32, 34, 34, 50, 350, 446

curve, if the order of its Jacobian is a product of an integer less than 2^{10} and a prime, the value of the integer is listed in the second column. Thus the orders of the Jacobians of those curves are divisible by primes greater than 2^{162}. In Table 2, we list values of u and v attained the best possible in the sense that the order of the Jacobian of the curve is two times of a prime: So, the probability to obtain the best possible curves in the above experiment is $\frac{13}{2000} = 0.0065$. By the prime number theorem, a "density" of primes around $A^2/2$ is approximately $1/\log_e(A^2/2) \approx 0.0084$. In the case $p = A - 677$, no best possible curve was found in the above computation. But further computation finds that the order of the Jacobian of the curve with $u = 935898796293578496671042 47$ and $v = 22435109140875626789358 13$ is a product of 2 and a prime

$$472920528303753106662785290707807991913732585053431 7.$$

This suggests that we can find a curve of the form $Y^2 = X^5 + uX^3 + vX$ for cryptographic use for any p. However, one might want to generate a random prime p and $u \in \mathbf{F}_p$, $v \in \mathbf{F}_p^\times$ for each curve.

Another observation is that curves with cofactor 6 and 12 were not found, where four curves with cofactor 18 were found. The author has no explanation of such phenomena.

The order of Jacobians of the curves $Y^2 = X^5 + uX^3 + vX$ seems less likely to be a product of a small positive integer and a large prime than a random integer. However, our experiments also suggests that our algorithm generates hyperelliptic curves with acceptable computational time.

We end this section with examples of curves to which we can apply Gaudry's efficient scalar multiplication [14]. Let $p = 2^{87} - 67$. Among 400 randomly generated pairs $(\alpha, \beta) \in \mathbf{F}_p^\times \times \mathbf{F}_p^\times$ such that $\alpha\beta$ is not a quadratic residue and that $\alpha \neq \pm\beta$, two of them gave the curve $Y^2 = X(X^2 - \alpha^2)(X^2 - \beta^2)$ with a cofactor 16. The corresponding values of u and v are

$$\begin{aligned}(u, v) = {} & (143809213492221778144040547, 131138969142842659104031301),\\ & (152044042900483697472576701, 438738171348065571223361 46).\end{aligned}$$

Table 2. Curves with cofactor 2

p	u, v, order/2 (= prime)
$A-23$	26278410876831238768152256, 86364989829465111812877054, 472920528303659680345114257217571460043665322659127
$A-23$	48005263478608513799676536, 41220251281438002920145925, 472920528303659680345114258064366296670183406802115 9
$A-23$	20282462437998363621453892, 68366944575985333923275 45, 472920528303659680345114258133732817650284170540220 7
$A-35$	67648337171838833749692452, 56901612904748554441926559, 472920528303659680345114138072169120188545555476779 1
$A-39$	14846780454565145653845797, 21699720008937250121391434, 472920528303659680345114102353627309819453876693252 7
$A-39$	33146413473211029791343648, 80307184653756356179241 26, 472920528303659680345114100348567188574110420995699 1
$A-39$	71798113541184209350130597, 32655964372648182436520 42, 472920528303659680345114098177535158246221087181274 3
$A-179$	57049947468040934287481804, 79903956336370051333994749, 472920528303659680345112737905752263561298973395143 1
$A-267$	36056874937457171776037216, 25118608280476680778431722, 472920528303659680345111869267728944477108168614507 1
$A-413$	83008004756000748681115585, 56563321965691537707290563, 472920528303561360104940130312551258067595321824591 7
$A-449$	28607196238761965671229454, 25170972284073882698894414, 472920528303673272123015906401321288517730330017407 3
$A-609$	95130756913811082727444951, 60322547687988038644971694, 472920528303771657998153663085486775700590453605511 3
$A-957$	10340770761867668646270127, 33230037969144840368622007, 472920528303608321884723156658609342418443268378843 7

Since all 2-torsion points are $\mathbf{F}_q$-rational, the cofactor 16 is the best possible value. We need further experiments and analysis to observe the probability of such curves being suitable for cryptography.

8 Security Considerations

We described an algorithm to generate a hyperelliptic curve $C/\mathbf{F}_q$ such that the order of Jacobian J of C is a product of a small number and a large prime. Since C is of the special form $Y^2 = X^5 + uX^3 + vX$, the curve C and its Jacobian J have some special structures. This might affect difficulties of the discrete log problems on $J(\mathbf{F}_q)$. We consider the problem.

First we note that the automorphism group of J is larger than that of a generic genus two hyperelliptic curve. One might think the larger automorphism group fasten the discrete log computation on $J(\mathbf{F}_p)$ by using the technique described in Duursma, Gaudry and Morain[7]. To some extent, it does. However, its influence is rather limited. Let us observe a reason. Let t be a time to perform one

multiplication on $\mathbf{F}_p$. Let G be a (not necessarily finite, commutative) group acting on $J(\mathbf{F}_q)$ from left. We assume that the multiplication by (-1)-map belongs to G and that, expect for the (-1)-times map, a computation of the action of an element of G needs more time than t. Put $N = \#J(\mathbf{F}_q)$ and $\overline{N} = \#(G \setminus J(\mathbf{F}_q))$ for simplicity. The above mentioned method uses a random walk on the coset space $G \setminus J(\mathbf{F}_q)$ while original Pollard ρ method uses a random walk on $J(\mathbf{F}_q)$. Thus in order to solve DLP on $J(\mathbf{F}_q)$, the expected number of walks is reduced by a factor of $(N/\overline{N})^{1/2}$. However, we need to consider some extra costs. At each random walk step, we need to find a representative of a G-orbit to test a match on $G \setminus J(\mathbf{F}_q)$. For $P \in J(\mathbf{F}_q)$, the representative is a point having some properties (e.g. having a minimal X-coordinate regarded as an integer) among the points in the orbit $G \cdot P$. The time for this is no less than $t\#(G \cdot P)/2$. In average, this is $\frac{tN}{2\overline{N}}$. Let T be a time to compute the next point (not an equivalence class) by one step walk on $J(\mathbf{F}_q)$. Thus, the method in [7] runs faster than original Pollard ρ, in average, by a factor of

$$\frac{T\sqrt{N}}{\sqrt{\overline{N}}\left(T + \frac{tN}{2\overline{N}}\right)} \leq \sqrt{\frac{T}{2t}},$$

regardless of G. In our case, we may assume that $T/2t \leq 2^{16}$, for example. (In reality, we can perform an addition on $J(\mathbf{F}_q)$ with less than 100 multiplications over $\mathbf{F}_q$ and q is, at most, a small power of p.) We have only to increase the size of q by four bits.

A more powerful attack to our curve would be the index calculus method due to Gaudry[15]. In short, the method solve the DLP on an elliptic curve defined over $\mathbf{F}_{q^n}$ whose heuristic time complexity (with respect to the number of group operations) is $\tilde{O}(q^{2-(2/n)})$ as $q \to \infty$ for fixed n. The O-constant depends on n. Generically, a polynomial of degree $2^{n(n-1)}$ appears during the computation.

Now, it is obvious that the DLP on $J(\mathbf{F}_q)$ is reduced to that on $E/\mathbf{F}_q(s)$, where E and s are defined as in Section 3. Recall that $[\mathbf{F}_q(s) : \mathbf{F}_q]$ is either two or four. So, the time complexity of Gaudry's index calculus method is $\tilde{O}(q)$ and $\tilde{O}(q^{3/2})$, according to $n = 2$ and 4, respectively. On the other hand, the order of $J(\mathbf{F}_q)$ is $O(q^2)$ and square root algorithms solve the DLP on $J(\mathbf{F}_q)$ with $O(q)$ group operations. Thus, Gaurdy's index calculus method does not provide an asymptotically faster attack than square root algorithms, at least for the case $q = p$. In case of $q > p$, we can consider the Weil restriction of E to a subfield of $\mathbf{F}_q$ with a lager n. When $[\mathbf{F}_q : \mathbf{F}_p]$ is even and $[\mathbf{F}_q(s) : \mathbf{F}_q] = 2$, Gaudry's index calculus with $n = 4$ runs moderate space complexity and its time complexity is $\tilde{O}(q^{3/4})$, at least asymptotically. This case must be avoided. In the other cases, the efficiency of Gaudry's index calculus is questionable due to its huge space complexity. It is probably prudent to take $q = p$ in cryptographic applications.

Remark 7. We also need further tests for suitability to hyperelliptic curve cryptosystem for general hyperelliptic curves. These includes (but not limited to) the following conditions. The minimal embedding degree (in the sense of Hitt[21]) should not be too small to avoid multiplicative DLP reduction by Frey and

Rück[10]. If M is close to $(\sqrt{q}-1)^2$ by some reason, the algorithm may return p as the largest prime factor. In this case, the input curve must not be used because it is vulnerable to the additive DLP reduction by Rück[34].

9 Conclusion

We presented an efficient algorithm to test whether a given hyperelliptic curve $Y^2 = X^5 + uX^3 + vX$ over $\mathbf{F}_q$ is suitable or not for cryptography and if suitable, the largest prime divisor of the number of $\mathbf{F}_q$-rational points of the Jacobian of the curve. This enables us to use a randomly generated hyperelliptic curve in the form which is supposed to be suitable for hyperelliptic curve cryptography. Although our family of curves is far more general than the curves given by binomials, it is still in a very special form. The difficulty of the discrete log problem on the simple but not absolutely simple Jacobian is open.

Acknowledgments

The major part of the work is performed during the author visited Prof. Steven Galbraith at Royal Holloway University. The author would like to thank their hospitality during his stay. He also would like to thank Steven Galbraith, Pierrick Gaudry, Florian Hess, Tanja Lange, Katsuyuki Takashima, Frederik Vercauteren and the anonymous reviewers for their helpful comments on this work.

References

1. Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P. Ann. of Math. 160, 781–793 (2004)
2. Anuradha, N.: Number of points on certain hyperelliptic curves defined over finite fields. Finite Fields Appl. 14, 314–328 (2008)
3. Berndt, B.C., Evans, R.J., Williams, K.S.: Gauss and Jacobi sums. John Wiley & Sons, Inc., New York (1998)
4. Cantor, D., Zassenhaus, H.: A new algorithm for factoring polynomials over finite fields. Math. Comp. 36, 587–592 (1981)
5. Cassels, J.W.S., Flynn, E.V.: Prolegomena to a middlebrow arithmetic of curves of genus 2. London Math. Soc. Lecture Note Series, vol. 230. Cambridge Univ. Press, Cambridge (1996)
6. Chebyshev, P.L.: Mémoire sur les nombres premiers. J. Math. Pures Appl. 17, 366–390 (1852) Œuvres, I-5
7. Duursma, I.M., Gaudry, P., Morain, F.: Speeding up the discrete log computation on curves with automorphisms. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 103–121. Springer, Heidelberg (1999)
8. Elkies, N.D.: Elliptic and modular curves over finite fields and related computational issues. In: Buell, D.A., Teitelbaum, J.T. (eds.) Computational perspectives on number theory, Chicago, IL (1995); AMS/IP Stud. Adv. Math., vol. 7, pp. 21–76. AMS, Providence, RI (1998)

9. Frey, G., Kani, E.: Curves of genus 2 covering elliptic curves and an arithmetical application. In: van der Geer, G., Oort, F., Steenbrink, J. (eds.) Arithmetic algebraic geometry, Texel (1989); Progress in Math., vol. 89, pp. 153–176. Birkhäuser Boston, Boston (1991)
10. Frey, G., Rück, H.-G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comp. 62, 865–874 (1994)
11. Furukawa, E., Kawazoe, M., Takahashi, T.: Counting points for hyperelliptic curves of type $y^2 = x^5 + ax$ over finite prime fields. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 26–41. Springer, Heidelberg (2004)
12. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (2001)
13. Gaudry, P.: An algorithm for solving the discrete log problem on hyperelliptic curves. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 19–34. Springer, Heidelberg (2000)
14. Gaudry, P.: Fast genus 2 arithmetic based on theta functions. J. Math. Cryptology 1, 243–265 (2007)
15. Gaudry, P.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. J. Symbolic Comput. (2008), doi:10.1016/j.jsc.2008.08.005
16. Gaudry, P., Schost, É.: On the invariants of the quotients of the Jacobian of a curve of genus 2. In: Bozta, S., Sphparlinski, I. (eds.) AAECC 2001. LNCS, vol. 2227, pp. 373–386. Springer, Heidelberg (2001)
17. Gaudry, P., Schost, É.: Construction of secure random curves of genus 2 over prime fields. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 239–256. Springer, Heidelberg (2004)
18. Gaudry, P., Schost, É.: Hyperelliptic point counting record: 254 bit Jacobian. Post to NMBRTHRY list (June 22, 2008)
19. Haneda, M., Kawazoe, M., Takahashi, T.: Suitable curves for genus-4 HCC over prime fields: point counting formulae for hyperelliptic curves of type $y^2 = x^{2k+1} + ax$. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 539–550. Springer, Heidelberg (2005)
20. Hess, F., Seroussi, G., Smart, N.P.: Two topics in hyperelliptic cryptography. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 181–189. Springer, Heidelberg (2001)
21. Hitt, L.: On the minimal embedding field. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 294–301. Springer, Heidelberg (2007)
22. Kedlaya, K.: Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. J. Ramanujan Math. Soc. 16, 323–338 (2001)
23. Lauder, A.G.B.: Rigid cohomology and p-adic point counting. J. Théor. Nombres Bordeaux 17, 169–180 (2005)
24. Leprévost, F., Morain, F.: Revêtements de courbes elliptiques à multiplication complexe par des courbes hyperelliptiques et sommes de caractéres. J. Number Theory 64, 165–182 (1997)
25. Lercier, R.: Finding Good Random Elliptic Curves for Cryptosystems Defined over $\mathbb{F}_{2^n}$. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 379–392. Springer, Heidelberg (1997)
26. Lercier, R., Lubicz, D.: A quasi quadratic time algorithm for hyperelliptic curve point counting. Ramanujan J. 12, 399–423 (2006)

27. Matsuo, K., Chao, J., Tsujii, S.: An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields. In: Fieker, C., Kohel, D.R. (eds.) ANTS 2002. LNCS, vol. 2369, pp. 461–474. Springer, Heidelberg (2002)
28. Maurer, M., Müller, V.: Finding the eigenvalue in Elkies' algorithm. Experimental Math. 10, 275–285 (2001)
29. Milne, J.S.: Abelian varieties. In: Cornell, G., Silverman, J.H. (eds.) Arithmetic Geometry, pp. 103–150. Springer, New York (1986)
30. Milne, J.S.: Jacobian varieties. In: Cornell, G., Silverman, J.H. (eds.) Arithmetic Geometry, pp. 167–212. Springer, New York (1986)
31. Paulhus, J.: Decomposing Jacobians of curves with extra automorphisms. Acta Arith. 132, 231–244 (2008)
32. Pila, J.: Frobenius maps of Abelian varieties and finding roots of unity in finite fields. Math. Comp. 55, 745–763 (1990)
33. Pohlig, S.C., Hellman, M.E.: An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Trans. Info. Theory 24, 106–110 (1978)
34. Rück, H.G.: On the discrete logarithm in the divisor class group of curves. Math. Comp. 68, 805–806 (1999)
35. Scholten, J.: Weil restriction of an elliptic curve over a quadratic extension. preprint, `http://homes.esat.kuleuven.be/~jscholte/`
36. Schoof, R.: Elliptic curves over finite fields and the computation of square roots mod p. Math. Comp. 44, 483–494 (1985)
37. Schoof, R.: Counting points on elliptic curves over finite fields. J. Théor. Nombres Bordeaux 7, 219–254 (1995)
38. Sutherland, A.V.: A generic apporach to searching for Jacobians. Math. Comp. 78, 485–507 (2009)
39. Takashima, K.: A new type of fast endomorphisms on Jacobians of hyperelliptic curves and their cryptographic application. IEICE Trans. Fundamentals E89-A, 124–133 (2006)
40. Vercauteren, F.: Advances in point counting. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) Advances in elliptic curve cryptography. London Math. Sco. Lecture Note Ser, vol. 317, pp. 103–132. Cambridge Univ. Press, Cambridge (2005)
41. Vercautern, F.: The SEA algorithm in characteristic 2, preprint (2000), `http://homes.esat.kuleuven.be/~fvercaut/papers/SEA.pdf`
42. von zur Gathen, J., Gerhard, J.: Modern computer algebra, 2nd edn. Cambridge UP, Cambridge (2003)
43. von zur Gathen, J., Shoup, V.: Computing Frobenius maps and factoring polynomials. Computational complexity 2, 187–224 (1992)

Verifiable Random Functions from Identity-Based Key Encapsulation*

Michel Abdalla[1], Dario Catalano[2,**], and Dario Fiore[2]

[1] CNRS–LIENS, Ecole Normale Supérieure, Paris, France
michel.abdalla@ens.fr
[2] Dipartimento di Matematica e Informatica, Università di Catania, Italy
{catalano,fiore}@dmi.unict.it

Abstract. We propose a methodology to construct verifiable random functions from a class of identity based key encapsulation mechanisms (IB-KEM) that we call VRF suitable. Informally, an IB-KEM is VRF suitable if it provides what we call *unique decryption* (i.e. given a ciphertext C produced with respect to an identity ID, all the secret keys corresponding to identity ID', decrypt to the same value, even if $ID \neq ID'$) and it satisfies an additional property that we call pseudorandom decapsulation. In a nutshell, pseudorandom decapsulation means that if one decrypts a ciphertext C, produced with respect to an identity ID, using the decryption key corresponding to any other identity ID' the resulting value looks random to a polynomially bounded observer. Interestingly, we show that most known IB-KEMs already achieve pseudorandom decapsulation. Our construction is of interest both from a theoretical and a practical perspective. Indeed, apart from establishing a connection between two seemingly unrelated primitives, our methodology is *direct* in the sense that, in contrast to most previous constructions, it avoids the inefficient Goldreich-Levin hardcore bit transformation.

1 Introduction

Verifiable Random Functions (VRFs for short) were introduced by Micali, Rabin and Vadhan [21]. Informally a VRF is something that behaves like a random function but also allows for efficient verification. More precisely, this means that associated with a secret key sk (the seed), there is a public key pk and a function F such that the following properties are satisfied. First, the function is efficiently computable, given sk, on any input. Second, having only pk and oracle access to the function, the value $F_{pk}(x) = y$ looks random to any polynomially bounded observer who did not query $F_{pk}(x)$ explicitly. Third, a proof $\pi_{pk}(x)$ that $F_{pk}(x) = y$ is efficiently computable knowing sk and efficiently verifiable knowing only pk.

VRFs turn out to be very useful in a variety of applications essentially because they can be seen as a compact commitment to an exponential number of

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* The full version of this paper is available at http://www.dmi.unict.it/~fiore

** Work partially done while visiting the computer science department at Ecole Normale Supérieure.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 554–571, 2009.

(pseudo)random bits. To give a few examples, Micali and Reyzin [22] show how to use VRFs to reduce to 3 the number of rounds of resettable zero knowledge proofs in the bare model. Micali and Rivest [23] described a very simple non interactive lottery system used in micropayment schemes, based on VRFs. Jarecki and Shmatikov [17] employed VRFs to build a verifiable transaction escrow scheme that preserves users anonymity while enabling automatic de-escrow. Liskov [18] used VRFs to construct updatable Zero Knowledge databases. In spite of their popularity VRFs are not very well understood objects. In fact, as of today, only four constructions are known, in the standard model [21,20,11,13]. The schemes given in [21,20] build VRFs in two steps. First they focus on constructing a *Verifiable Unpredictable Function* (VUF). Informally a VUF is a function that is hard to compute but whose produced outputs do not necessarily look random. Next they show how to convert a VUF into a VRF using the Goldreich-Levin [15] theorem to "extract" random bits. Unfortunately, however, the VRF resulting from this transformation is very inefficient and, furthermore, it looses a quite large factor in its exact security reduction. This is because, the transformation involves several steps, all rather inefficient. First one uses the Goldreich Levin theorem [15] to construct a VRF with very small (i.e. slightly super polynomial in the security parameter) input space and output size 1. Next, one iterates the previous step in order to amplify the output size to (roughly) that of the input. Then, using a tree based construction, one iterates the resulting function in order to get a VRF with unrestricted input size and finally one evaluates the so obtained VRF several times in order to get an output size of the required length.

The constructions given in [11,13], on the other hand, are direct, meaning with this that they manage to construct VRF without having to resort to the Goldreich Levin transform. The VRF presented in [11] is based on a "DDH-like" assumption that the author calls *sum-free decisional Diffie-Hellman* (sf-DDH). This assumption is similar to that one employed by Naor-Reingold [24] to construct PRFs, with the difference that it applies an error correcting code C to the input elements in order to compute the function. The specific properties of the employed encoding allow for producing additional values that can be used as proofs. This construction is more efficient than [21,20] in the sense that it does not need the expensive Goldreich Levin transform. Still it has some efficiency issues as the size of the produced proofs and keys is linear in the input size. Dodis [11] also adapts this construction to provide a *distributed* VRF, that is a standard VRF which can be computed in a distributed manner.

The scheme proposed by Dodis and Yampolskiy [13], on the other hand, is more attractive, at least from a practical point of view, as it provides a simple implementation of VRFs with short (i.e. constant size) proofs and keys. It is interesting to note that, even though the latter construction is far more efficient than previous work, it builds upon a similar approach. Basically, the construction in [13] works in two steps. First they consider a simple VUF (which is basically Boneh Boyen [3] weakly secure signature scheme) that is secure for slightly superpolynomially sized input spaces. Next, rather than resorting to the Godreich Levin [15] hardcore bit theorem to convert it into a VRF, they show

how to modify the original VUF in order to make it a VRF, under an appropriate decisional assumption.

From the discussion above, it seems clear that, with the possible exception of [11], all known constructions of verifiable random functions, follow similar design criteria. First one builds a suitable VUF and then transforms it into a VRF by either using the Goldreich Levin transform, or via some direct, ad hoc, modifications of the original VUF. The main drawback of this approach is that, once a good enough VUF is found, one has to either be able to make it a VRF directly or accept the fact that the VRF obtained from the Goldreich Levin transform is not going to be a practical one. Thus it seems very natural to ask if there are alternative (and potentially more efficient) ways that allow to construct VRFs directly, without needing to resort to the two steps methodology sketched above.

Our Contribution. In this paper we show how to construct VRF from a class of identity based encryption (IBE) schemes [26] that we call *VRF suitable*. Roughly speaking an identity based encryption scheme, is an asymmetric encryption scheme where the public key can be an arbitrary string. Such schemes consists of four algorithms. A Setup algorithm, that generates the system common parameters as well as a master key msk; a key derivation algorithm that uses the master secret key to generate a private key d_{sk} corresponding to an arbitrary public key string ID (the identity); an encryption algorithm that encrypts messages using the public key ID and a decryption algorithm that decrypts ciphertexts using the corresponding private key.

Informally an IBE is said to be VRF suitable if the following conditions are met. First, the scheme has to provide *unique decryption*. This means that, given a ciphertext C produced with respect to some arbitrary identity ID, all the secret keys corresponding to any other identity ID' decrypt to the same value (i.e. even if $ID' \neq ID$). Second, the Key Encapsulation Mechanism (KEM) associated with the IBE (see below for a definition of key encapsulation mechanism) has to provide what we call *pseudorandom decapsulation*. Very informally, pseudorandom decapsulation means that if C is an encapsulation produced using some identity ID, the "decapsulated" key should look random even if the decapsulation algorithm is executed using the secret key corresponding to any other identity $ID^* \neq ID$. Having a scheme that achieves pseudorandom decapsulation may seem a strong requirement at first. We argue that it is not, as basically all currently known secure (in the standard model) IBE schemes *already* provide pseudorandom decapsulation.

Our result is of interest both from a theoretical and a practical point of view. Indeed, apart from establishing a connection between two seemingly unrelated primitives, our method is direct, in the sense that it allows to build a VRF from a VRF suitable IBE without having to resort to the inefficient Goldreich Levin transform. Moreover, the reduction is tight. This means that, once an efficient VRF suitable IBE is available, this leads to an equally efficient VRF, with no additional security loss. Furthermore, our constructions immediately allow

for efficient distributed VRFs as long as a distributed version of the underlying encryption scheme is available (which is the case for most schemes used in practice).

As a second contribution of this paper, we investigate on the possibility of implementing VRF suitable IBEs. Toward this goal, we first show how to construct a VRF suitable IB KEM from the Sakai-Kasahara IB KEM [25]. Interestingly, the resulting VRF turns out to be very similar to the Dodis-Yampolskiy VRF [13], thus showing that the latter construction can actually be seen as a special case of our general methodology. Next, we propose a new implementation of VRF suitable IB KEM inspired (but more efficient) by Lysyanskaya's VRF [20] (which in turn builds from the Naor Reingold's PRF [24]). The proposed scheme can be proved secure under the assumed intractability, in bilinear groups, of the decisional ℓ-th weak Bilinear Diffie Hellman Inversion problem (decisional ℓ-wBDHI* for short) introduced by Boneh, Boyen and Goh [4]. Interestingly, even though the decisional ℓ-wBDHI* assumption is asymptotic in nature, the ℓ parameter does not need to be too large in order for our security proof to go through. This is because it directly affects only the size of the space of valid identities *but not* the number of adversarial queries allowed in the security reduction[1] (as opposed to most known proofs using asymptotic assumptions). This means that in practice it is enough to assume the decisional ℓ-wBDHI* assumption to hold only for reasonably small values of ℓ (such as $\ell = 160$ or $\ell = 256$).

IBEs and Digital Signatures. Naor pointed out (see [5]) that a fully secure identity based encryption scheme can be transformed into a secure signature scheme as follows. One sets the message space as the set I of valid identities of the IBE. To sign $m \in I$ one executes the key derivation algorithm on input m, and outputs d_{sk} as the signature. A signature on m is verified by encrypting a random M with respect to the identity m, and then by checking that decrypting the resulting ciphertext one gets back M. Thus if one considers an IBE with unique key derivation (i.e. where for each identity one single corresponding decryption key can be computed) the methodology sketched above leads to a secure *unique* digital signature scheme (i.e. a digital signature scheme for which each message admits one single valid signature). Since secure unique signatures are, by definition, verifiable unpredictable functions, at first glance our construction might seem to (somewhat) follow from Naor's remark. We argue that this does not seem to be the case for two reasons. First, our construction does not require the underlying IB-KEM to have unique key derivation, but only to provide unique decryption. Clearly the former property implies the latter, but there is no reason to exclude the possibility of constructing a scheme realizing unique decryption using a randomized key derivation procedure. Second, a crucial requirement for Naor's transformation to work is that the original IBE is actually fully secure. A VRF-suitable IBE, on the other hand, is required to be secure only in a much weaker sense (that we call *weak selective* ID security).

[1] Here by not affecting the number of adversarial queries we mean that ℓ grows linearly with respect to the identity space but only logarithmically with respect to the number of adversarial queries.

Other related Work. As pointed out above the notion of VRF is related to the notion of unique signatures. Unique signatures were introduced by Goldwasser and Ostrovsky [16] (they called them invariant signatures). The only known constructions of unique signatures in the plain model (i.e. without common parameters or random oracles) are due to Micali, Rabin and Vadhan [21], to Lysyanskaya [20] and to Boneh and Boyen [3]. In the common string model, Goldwasser and Ostrovsky [16] also showed that unique signatures require the same kind of assumptions needed to construct non interactive zero knowledge.

Dodis and Puniya in [12] address the problem of constructing Verifiable Random Permutations from Verifiable Random Functions. They define VRPs as the verifiable analogous of pseudorandom permutations. In particular they point out that the technique of Luby-Rackoff [19] (for constructing PRPs from PRFs) cannot be applied in this case. This is due to the fact that VRP proofs must reveal the VRF outputs and proofs of the intermediate rounds. In their paper they show a construction in which a super-logarithmic number of executions of the Feistel transformation suffices to build a VRP.

More recently Chase and Lysyanskaya [8] introduced the notion of simulatable VRF. Informally a simulatable VRF is a VRF with the additional property that proofs can be simulated, meaning with this that a simulator can fake proofs showing that the value of $F_{sk}(x)$ is y for any y of its choice. Simulatable VRFs can be used to provide a direct transformation from single theorem non interactive zero knowledge to multi theorem NIZK and work in the common reference string model.

2 Preliminaries

Before presenting our results we briefly recall some basic definitions. In what follows we will denote with k a security parameter. The participants to our protocols are modeled as probabilistic Turing machines whose running time is bounded by some polynomial in k. Denote with $\mathbb{N}$ the set of natural numbers and with $\mathbb{R}^+$ the set of positive real numbers. We say that a function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is negligible if and only if for every polynomial $P(k)$ there exists an $k_0 \in \mathbb{N}$ such that for all $k > k_0$ $\epsilon(k) < 1/P(k)$. If A is a set, then $a \stackrel{\$}{\leftarrow} A$ indicates the process of selecting a at random and uniformly over A (which in particular assumes that A can be sampled efficiently).

Verifiable Random Functions Verifiable Random Functions (VRFs for short) were introduced by Micali, Rabin and Vadhan [21]. Intuitively, a VRF is something that behaves like a pseudorandom function, but also allows for a proof of its output correctness. More formally, a VRF is a triplet of algorithms $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Func}, \mathsf{V})$ providing the following functionalities. The key generation algorithm Gen is a probabilistic algorithm that takes as input the security parameter and produces a couple of matching public and private keys (vpk, vsk). The deterministic algorithm Func, on input the secret key vsk and the input x to the VRF, computes $(\mathsf{F}_{vsk}(x), \mathsf{Prove}_{vsk}(x))$, where $\mathsf{F}_{vsk}(x)$ is the value of the VRF and $\mathsf{Prove}_{vsk}(x)$ its proof of correctness. The verification algorithm V takes

as input (vpk, x, v, π) and outputs a bit indicating whether or not π is a valid proof that $\mathsf{F}_{vsk}(x) = v$.

Let $a : \mathbb{N} \to \mathbb{N} \cup \{*\}$ and $b : \mathbb{N} \to \mathbb{N}$ be functions computable in polynomial time (in k). Moreover we assume that $a(k)$ and $b(k)$ are bounded by a polynomial in k, except if a takes the value $*$ (in this case we simply assume that the VRF can take inputs of arbitrary length). Formally, we say that $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Func}, \mathsf{V})$ is a VRF of input length $a(k)$ and output length $b(k)$, if the following conditions are met.

Domain Range Correctness. For all $x \in \{0,1\}^{a(k)}$ it has to be the case that $\mathsf{F}_{vsk}(x) \in \{0,1\}^{b(k)}$. We require this condition to hold with overwhelming probability (over the choices of (vpk, vsk)).

Provability. For all $x \in \{0,1\}^{a(k)}$ if $\mathsf{Prove}_{vsk}(x) = \pi$ and $\mathsf{F}_{vsk}(x) = v$ then $\mathsf{V}(vpk, x, v, \pi) = 1$. We require this condition to hold with overwhelming probability (over the choices of (vpk, vsk) and the coin tosses of V).

Uniqueness. No values $(vpk, x, v_1, v_2, \pi_1, \pi_2)$ can satisfy (unless with negligible probability over the coin tosses of V) $\mathsf{V}(vpk, x, v_1, \pi_1) = \mathsf{V}(vpk, x, v_2, \pi_2) = 1$, when $v_1 \neq v_2$.

Pseudorandomness. For all probabilistic polynomial time adversaries $A = (A_1, A_2)$ we require that

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} (vpk, vsk) \xleftarrow{\$} \mathsf{Gen}(1^k);\ (x, \omega) \leftarrow A_1^{\mathsf{Func}(\cdot)}(vpk) \\ b \xleftarrow{\$} \{0,1\};\ v_0 \leftarrow \mathsf{F}_{vsk}(x);\ v_1 \xleftarrow{\$} \{0,1\}^{b(k)} \\ b' \leftarrow A_2^{\mathsf{Func}(\cdot)}(\omega, v_b) \end{array} \right] \leq \frac{1}{2} + \epsilon(k)$$

where the notation $A^{\mathsf{Func}(\cdot)}$ indicates that A has oracle access to the algorithm Func. In order to make this definition sensible, we impose that A cannot query the oracle on input x.

Remark 1. One might consider a relaxation of the pseudorandomness property in which the adversary is required to commit ahead of time (i.e. before seeing the public key) to the input value it intends to attack. We call *selective-VRF* a VRF that satisfies this weaker pseudorandomness[2].

ID BASED ENCRYPTION An identity based encryption scheme is a tuple of algorithms $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ providing the following functionality. The trusted authority runs Setup, on input 1^k, to generate a master key pair (mpk, msk). Without loss of generality we assume that the public key mpk specifies a message space $\mathcal{M}$ and a value n (polynomial in the security parameter) indicating the length of each identity. It publishes the master public key mpk and keeps the master secret key msk private. When a user with identity ID wishes to become part of the system, the trusted authority distributor generates a user decryption key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$, and sends this key over a secure and authenticated channel to the user. To send an encrypted message m to the user

[2] For lack of space we defer a more formal definition of this notion to the full version of this paper.

with identity ID, the sender computes the ciphertext $C \xleftarrow{\$} \mathsf{Enc}(mpk, ID, m)$, which can be decrypted by the user as $m \leftarrow \mathsf{Dec}(d_{ID}, C)$.

Boneh and Franklin [5] formally defined the notion of security for identity based encryption schemes. In particular they defined chosen plaintext security against adaptive chosen identity attack. Intuitively, such a notion, captures the requirement that security should be preserved even when facing an adversary who is allowed to choose the identity it wishes to attack. Later, Canetti, Halevi, and Katz [7] introduced a weaker notion of security in which the adversary is required to commit ahead of time (i.e. before the parameters of the scheme are made public) to the identity it intends to attack. A scheme meeting such a weaker security requirement is said selective ID, chosen plaintext secure IBE (IND-sID-CPA).

In this paper we introduce a new notion of security for IBE schemes that we call *weak selective ID security*. More precisely, we define weak selective ID security as the full fledged selective case with the exception that here the challenge identity is chosen by the challenger and given in input to the adversary. Clearly, this notion is weaker with respect to selective ID security as it is easy to see that the latter implies the former.

Identity Based Key Encapsulation. An identity-based key encapsulation mechanism (IB-KEM) scheme allows a sender and a receiver to agree on a random session key K in such a way that the sender can create K from public parameters and receiver identity and the receiver can recover K using his secret key. This notion, in the context of identity-based encryption, was first formalized by Bentahar et al. [1].

An IB-KEM scheme is defined by four algorithms:

- $\mathsf{Setup}(1^k)$ is a probabilistic algorithm that takes in input a security parameter k and outputs a master public key mpk and a master secret key msk.
- $\mathsf{KeyDer}(msk, ID)$ The key derivation algorithm uses the master secret key to compute a secret key sk_{ID} for identity ID.
- $\mathsf{Encap}(mpk, ID)$ The encapsulation algorithm computes a random session key K and a corresponding ciphertext C encrypted under the identity ID.
- $\mathsf{Decap}(C, sk_{ID})$ allows the possessor of a secret key sk_{ID} to decapsulate C to get back a session key K. We denote by $\mathcal{K}$ the session key space.

For correctness it is required that $\forall k \in \mathbb{N}, ID \in \mathcal{ID}, (C, K) \xleftarrow{\$} \mathsf{Encap}(mpk, ID)$ the following probability holds for all possible $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$:

$$Pr[\mathsf{Decap}(C, \mathsf{KeyDer}(msk, ID)) = K] = 1$$

Here we define the notion of *weak selective ID security* for IB-KEM schemes. Let $\mathcal{IBKEM}$ be a IBE scheme with key encapsulation mechanism. Then $\mathcal{IBKEM}$ is *weakly selective ID secure against adaptively-chosen plaintext attacks* (wsIB-KEM-CPA) if there exists no polynomially bounded adversary $\mathcal{A}$ with non negligible advantage against the Challenger in the following game:

Setup. In this phase the challenger selects a challenge identity ID^* (according to an arbitrary distribution) and runs $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$. Then it

computes $(C^*, K^*) = \mathsf{Encap}(mpk, ID^*)$ and flips a binary coin $b \xleftarrow{\$} \{0,1\}$. Then it sets $\bar{K} = K^*$ if $b = 0$, otherwise it picks a random key $\bar{K} \xleftarrow{\$} \mathcal{K}$. Finally it runs $\mathcal{A}$ on input $(mpk, ID^*, C^*, \bar{K})$ and keeps msk for itself.

Key derivation queries. The adversary is allowed to ask key derivation queries for an arbitrary (but polynomial) number of adaptively chosen identities different from ID^*.

Guess. In the end of this game $\mathcal{A}$ outputs b' as its guess for b.

The adversary wins if $b' = b$. We formally define the advantage of $\mathcal{A}$ against $\mathcal{IBKEM}$ in the above game as

$$\mathbf{Adv}^{\text{wsIB-KEM-CPA}}_{\mathcal{IBKEM},\mathcal{A}}(k) = \left| Pr[b = b'] - \frac{1}{2} \right|$$

where the probability is taken over the coin tosses of the challenger and the adversary.

VRF-SUITABLE IB-KEMs. Our VRF construction relies on a special class of identity based key encapsulation mechanisms that we call *VRF suitable*. In particular, a VRF suitable IB-KEM is defined by the following algorithms

- $\mathsf{Setup}(1^k)$ is a probabilistic algorithm that takes in input a security parameter k and outputs a master public key mpk and a master secret key msk.
- $\mathsf{KeyDer}(msk, ID)$ The key derivation algorithm uses the master secret key to compute a secret key sk_{ID} for identity ID and some auxiliary information aux_{ID} needed to correctly encapsulate and decapsulate the key.
- $\mathsf{Encap}(mpk, ID, aux_{ID})$ The encapsulation algorithm computes a random session key K, using (mpk, ID, aux_{ID}). Moreover it uses (mpk, ID) to computes a ciphertext C encrypted under the identity ID. Notice that aux_{ID} is required to compute K but not to compute C.
- $\mathsf{Decap}(C, sk_{ID}, aux_{ID})$ allows the possessor of sk_{ID} and aux_{ID} to decapsulate C to get back a session key K. We denote by $\mathcal{K}$ the session key space.

Remark 2. Notice that the description above differs from the one given for basic IB-KEM in that here we require the encapsulation and decapsulation mechanism to use some auxiliary information aux_{ID}, produced by KeyDer, to work correctly. Clearly if one sets $aux_{ID} = \perp$ one goes back to the original description. Thus the new paradigm is slightly more general as it allows to consider encapsulation mechanism where everybody can compute the ciphertext but only those knowing the aux_{ID} information can compute the key. Notice however that aux_{ID} does not allow, by itself, to decapsulate. In some sense, this auxiliary information should be seen as a value that completes the public key (rather than something that completes the secret key) [3]. Even though such a syntax may look totally meaningless in the standard public key scenario, it turns out to be extremely useful (see below) in our context.

[3] In fact this auxiliary information is not required to be kept secret in our constructions since the adversary can in principle obtain its value for any identity of its choice including the challenge identity (see definition of pseudorandom decapsulation).

Moreover, the IB-KEM has to satisfy the following properties:

1. **Unique decryption.** Let ID_0 be any valid identity and C a ciphertext encrypted under ID_0. We require that no valid identity ID can satisfy (unless with negligible probability) $\mathsf{Decap}(C, sk'_{ID}, aux_{ID}{}') \neq \mathsf{Decap}(C, sk''_{ID}, aux_{ID}{}'')$, where $(sk'_{ID}, aux_{ID}{}') \leftarrow \mathsf{KeyDer}(msk, ID)$, $(sk''_{ID}, aux_{ID}{}'') \leftarrow \mathsf{KeyDer}(msk, ID)$
2. **Pseudorandom decapsulation.** Let C be an encapsulation produced using identity ID_0, we require the session key to look random even if the decapsulation algorithm is executed using the secret key corresponding to any other $\overline{ID}$. More formally, we define the following experiment, for a polynomially bounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Experiment $\mathbf{Exp}^{\text{IB-KEM-RDECAP}}_{\mathcal{IBKEM},\mathcal{A}}(k)$
$(mpk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
Choose $ID_0 \in \mathcal{ID}$ (according to any arbitrary distribution)
$C^* \xleftarrow{\$} \mathsf{Encap}(mpk, ID_0)$
$(\overline{ID}, st) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{KeyDer}(\cdot)}(mpk, C^*, ID_0)$
$(aux_{\overline{ID}}, sk_{\overline{ID}}) \xleftarrow{\$} \mathsf{KeyDer}(msk, \overline{ID})$
$b \xleftarrow{\$} \{0,1\}$; $K_0 \xleftarrow{\$} \mathsf{Decap}(C^*, sk_{\overline{ID}}, aux_{\overline{ID}})$; $K_1 \xleftarrow{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{KeyDer}(\cdot)}(st, K_b, aux_{\overline{ID}})$
If $b' = b$ then return 1, else return 0

With $\mathcal{A}^{\mathsf{KeyDer}(\cdot)}$ we denote that an algorithm $\mathcal{A}$ has oracle access to the key derivation algorithm. Let $\mathcal{ID}$ denote identity space, i.e. the space from which the adversary (and everybody else) is allowed to choose the identities. In the experiment $\mathbf{Exp}^{\text{IB-KEM-RDECAP}}_{\mathcal{IBKEM},\mathcal{A}}$ we need the following restrictions:

- the identity $\overline{ID}$ output by $\mathcal{A}_1$ should not be asked before;
- $\mathcal{A}_2$ is not allowed to query the oracle on $\overline{ID}$.

We define the advantage of $\mathcal{A}$ in the IB-KEM-RDECAP experiment as

$$\mathbf{Adv}^{\text{IB-KEM-RDECAP}}_{\mathcal{IBKEM},\mathcal{A}}(k) = \left| Pr\left[\mathbf{Exp}^{\text{IB-KEM-RDECAP}}_{\mathcal{IBKEM},\mathcal{A}}(k) = 1\right] - \frac{1}{2}\right|.$$

$\mathcal{IBKEM}$ has pseudorandom decapsulation if for any polynomially bounded adversary $\mathcal{A}$ the advantage $\mathbf{Adv}^{\text{IB-KEM-RDECAP}}_{\mathcal{IBKEM},\mathcal{A}}(k)$ is a negligible function in k.

Remark 3. Requiring that an IB-KEM provides pseudorandom decapsulation might seem a very strong requirement at first. We argue that it is not, at least if the known constructions of IB-KEMs are considered. Indeed, all currently known schemes which are IND-CPA secure (but not IND-CCA secure) in the standard model *already* have this property (see the full version of the paper for details).

3 The Construction

In this section we show our construction of Verifiable Random Functions from a VRF-suitable IB-KEM $\mathcal{IBKEM} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap})$. Let $\mathcal{ID}$ be the identity space, $\mathcal{K}$ the session key space and $\mathcal{SK}$ the secret key space. Then we construct $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Func}, \mathsf{V})$ which models a function from input space $\mathcal{ID}$ to output space $\mathcal{K}$.

$\mathsf{Gen}(1^k)$ runs $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$, chooses an arbitrary identity $ID_0 \in \mathcal{ID}$ and computes $C_0 \leftarrow \mathsf{Encap}(mpk, ID_0)$. Then it sets $vpk = (mpk, C_0)$ and $vsk = msk$.

$\mathsf{Func}_{vsk}(x)$ computes $\pi_x = (sk_x, aux_x) = \mathsf{KeyDer}(msk, x)$ and $y = \mathsf{Decap}(C_0, \pi_x)$. It returns (y, π_x) where y is the output of the function and π_x is the proof.

$\mathsf{V}(vpk, x, y, \pi_x)$ first checks if π_x is a valid proof for x in the following way. It computes $(C, K) = \mathsf{Encap}(mpk, x, aux_x)$ and checks if $K = \mathsf{Decap}(C, \pi_x)$. Then it checks the validity of y by testing if $\mathsf{Decap}(C_0, \pi_x) = y$. If both the tests are true, then the algorithm returns 1, otherwise it returns 0.

Now we prove that the proposed construction actually realizes a secure VRF.

Theorem 1. *Assume $\mathcal{IBKEM}$ is a VRF Suitable IB-KEM scheme, as described in section 2 then the construction given above is a verifiable random function.*

Proof. According to the definition given in section 2, we prove that $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Func}, \mathsf{V})$ is a verifiable random function by showing that it satisfies all the properties. Domain range correctness and provability trivially follow from the IB-KEM scheme correctness. Since $\mathcal{IBKEM}$ has unique decryption the uniqueness property is satisfied for construction of VRF. To prove the residual pseudorandomness we assume there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the residual pseudorandomness of VRF with non-negligible probability $\frac{1}{2} + \epsilon(k)$. Then we can build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ which has non-negligible advantage $\epsilon(k)$ in the IB-KEM-RDECAP game.

$\mathcal{B}_1$ receives in input from its Challenger the public key mpk and a ciphertext C_0^*. It sets $vpk = (mpk, C_0^*)$ and runs $\mathcal{A}_1(vpk)$. The adversary $\mathcal{A}$ is allowed to make queries to the function oracle $\mathsf{Func}(\cdot)$. $\mathcal{B}$ simulates this oracle in the following way. Given input $x \in \mathcal{ID}$, it queries the key derivation oracle on x. It obtains sk_x and returns $(\mathsf{Decap}(C_0^*, sk_x), sk_x)$ to the adversary. When $\mathcal{A}_1$ outputs an element $\bar{x}$, $\mathcal{B}_1$ gives the same element to its challenger. Thus the challenger produces K^*, which is either the decapsulation of C_0^* with $sk_{\bar{x}}$ or a random element of $\mathcal{K}$, and gives it to $\mathcal{B}_2$. Then $\mathcal{B}_2$ runs $b' \leftarrow \mathcal{A}_2(st, K^*)$ and outputs b' to the Challenger.

Since the simulation is perfect, if $\mathcal{A}$ outputs $b' = b$ with probability $\frac{1}{2} + \epsilon(k)$, then $\mathcal{B}$'s advantage is exactly $\epsilon(k)$.

Notice that, when describing the notion of VRF suitable IB-KEM, we did not expect the underlying scheme to meet any additional security requirement. With the following theorem (whose proof is deferred to the full version of this paper) we show that, indeed, a necessary condition, in order for an IB-KEM to be VRF suitable, is that it is secure only in a weak selective sense.

Theorem 2. *Let $\mathcal{IBKEM}$ be a VRF Suitable IB-KEM, then it is also a weakly selective secure IB-KEM (in the sense of the definition given in section 2).*

4 VRF Suitable IBEs

In this section we describe our constructions of Verifiable Random functions from VRF suitable encryption schemes. In particular, in light of the results presented in section 3, we focus on constructing VRF suitable IB-KEM schemes.

We start by describing, in section 4.1, a VRF from the Sakai-Kasahara [25] IB-KEM. Interestingly, the proposed VRF is basically the same as the VRF proposed by Dodis and Yampolskiy [13], thus showing that their construction can be seen as a special case of our general paradigm.

Next we present, in section 4.2, a new construction of VRF suitable IB-KEM from an assumption related to the ℓ-Bilinear Diffie Hellman Inversion assumption (see [2]), that is known as the decisional ℓ-weak Bilinear Diffie Hellman Inversion assumption (decisional ℓ-wBDHI*, following the acronym used in [4]). The decisional ℓ-wBDHI* was introduced by Boneh, Boyen and Goh in [4] and it (informally) states that given $g^b, g^c, g^{b^2}, ..., g^{b^\ell}$, the quantity $e(g,g)^{b^{\ell+1}c}$ should remain indistinguishable from random to any polynomially bounded adversary. The assumption is related to the ℓ bilinear Diffie Hellman Inversion assumption (ℓ-BDHI), in the sense that the former is known to hold in all those groups where the latter holds, but the converse is not known to be true. Interestingly, in order for our construction to work, the ℓ parameter does not need to be too large. This is because it only limits to 2^ℓ the size of the space of valid identities but it does not affect in any other way the number of adversarial queries allowed in the security proof (as in most known proofs using q-type assumptions). Said in a different way, ℓ is required to grow only in a logarithmic way (rather than linearly) with respect to the number of adversarial queries allowed. This means that it is enough to assume that the ℓ-wBDHI* assumption holds only for rather small values of ℓ (i.e. $\ell = 160$ or $\ell = 256$).

As a final note we mention that, in principle, one could construct a VRF from Boneh Franklin's IBE. Indeed, in the full version of this paper, we show that the KEM version of the scheme is actually a VRF suitable IB-KEM, under the decisional Bilinear Diffie Hellman assumption. This construction, however, is of very limited interest, since the proof holds in the random oracle model.

4.1 Sakai-Kasahara VRF

We briefly recall the KEM version of the Sakai-Kasahara IBE scheme (`SK` for short) [25]. This scheme relies on the q-decisional Bilinear Diffie-Hellmann Inversion assumption (DBDHI for short), introduced by Boneh and Boyen in [2]. Informally, the DBDHI assumption in bilinear group G of prime order p states that, for every PPT algorithm $\mathcal{A}$ and for a parameter q, $\mathcal{A}$ has negligible probability into distinguishing $e(g,g)^{1/x} \in G_T$ from a random one after seeing $(g, g^x, g^{(x^2)}, \cdots, g^{(x^q)})$. If we suppose that $\mathcal{G}(1^k)$ is a bilinear group generator

which takes in input a security parameter k, then (asymptotically) the DBDHI assumption holds for $\mathcal{G}$ if $\mathcal{A}$'s probability of success is negligible in k, for any q polynomial in k.

- $\mathsf{Setup}(1^k)$ runs $\mathcal{G}(1^k)$ to obtain the description of the groups G, G_T and of a bilinear map $e : G \times G \to G_T$. The description of G contains a generator $g \in G$. Then the algorithm picks a random $s \xleftarrow{\$} \mathbb{Z}_p$ and sets $h = g^s$, $mpk = (g, h)$, $msk = s$.
- $\mathsf{KeyDer}(msk, ID)$ We assume $ID \in \mathbb{Z}_p$. The key derivation algorithm constructs the secret key $sk_{ID} = g^{\frac{1}{s+ID}}$.
- $\mathsf{Encap}(mpk, ID)$ The encapsulation algorithm picks a random $t \xleftarrow{\$} \mathbb{Z}_p^\star$ and computes a random session key $K = e(g, g)^t$ and a corresponding ciphertext $C = (g^s g^{ID})^t$.
- $\mathsf{Decap}(C, sk_{ID})$ the decapsulation algorithm uses the secret key sk_{ID} to compute a session key K from a ciphertext C as follows: $K = e(C, sk_{ID})$.

First notice that, assuming $aux_{ID} = \perp \ \forall ID$, the above description fits our syntax of VRF suitable IB-KEMs. Now we prove (for lack of space the actual proof appears in the full version of this paper) that the Sakai-Kasahara IB-KEM scheme can be used to construct a VRF (i.e. that it actually provides unique decryption and pseudorandom decapsulation). In particular, the resulting VRF can only support superpolynomially-sized (in the security parameter) input space. Notice that all known constructions of VRF made the same assumption.

Theorem 3. *Assuming that the DBDHI assumption holds in a bilinear group G, then the Sakai-Kasahara IBE scheme [25] is a VRF-suitable IBE.*

Similarity with the Dodis-Yampolskiy VRF. Now we show that the Dodis-Yampolskiy VRF [13] (that we briefly recall in appendix A) can be seen as a special instantiation of the construction given above. Indeed, theorem 3 leads to the following VRF.

- $\mathtt{Gen}(1^k)$ Runs $\mathcal{G}(1^k)$ to obtain the description of the groups G, G_T and of a bilinear map $e : G \times G \to G_T$. The description of G contains a generator $g \in G$. Then the algorithm picks random $s, t \xleftarrow{\$} \mathbb{Z}_p$ and sets $h = g^s$, $C_0 = h^t$, $vpk = (g, h, C_0)$, $vsk = s$.
- $\mathtt{Func}_{vsk}(x)$ Let $\mathtt{Func}_{vsk}(x)$=$(F_{vsk}(x), \pi_{vsk}(x))$. One sets $\mathtt{Func}_{vsk}(x)$= $e(C_0, sk_x)$ $= e(g, g)^{(st)/(s+x)}$ as the VRF output and $\pi_{vsk}(x) = \mathsf{KeyDer}(x) = g^{1/(s+x)}$ as the proof of correctness.
- $\mathtt{V}(vpk, x, y, \pi_x)$ To verify whether y was computed correctly, one starts by running the Encap algorithm on input (vpk, x). Encap chooses $\omega \xleftarrow{\$} \mathbb{Z}_p$ and then computes $K \leftarrow e(g, g)^\omega$ and $C = (hg^x)^\omega$. Then one checks that $K = \mathsf{Decap}(C, \pi_x) = e((g^x \cdot h)^\omega, \pi_x)$ and $y = \mathsf{Decap}(C_0, \pi_x) = e(h^t, \pi_x)$.

Thus by setting $t = s^{-1} \bmod p$ and $\omega = 1$, the construction above can be optimized to get exactly the Dodis Yampolskiy VRF.

4.2 The New Construction

In this section we propose a new construction of VRF suitable IB-KEM from the (conjectured) computational intractability of the decisional weak ℓ-Bilinear Diffie-Hellman Inversion problem (see below for a formal description). The new scheme is inspired from Lysyanskaya [20] VRF in that the validity of each new auxiliary information aux_{ID} (required to compute the session key) is verified by exploiting the DDH-CDH separation in bilinear groups. The new scheme however is more efficient as it leads to a VRF directly (i.e. rather than having to construct a unique signature scheme first) and does not require error correcting codes.

Decisional weak ℓ-Bilinear Diffie Hellman Inversion Problem [4]. The decisional ℓ-wBDHI* problem in G is defined as follows. Let G be a bilinear group of prime order p and g a generator of G. Given $g^b, g^c, g^{b^2}, ..., g^{b^\ell}$, we say that an algorithm $\mathcal{A}$ has advantage ϵ in solving decisional ℓ-wBDHI* in G if

$$\Pr[\mathcal{A}(g^c, g^b, g^{b^2}, ..., g^{b^\ell}, e(g,g)^{b^{\ell+1}c}) = 1] - \Pr[\mathcal{A}(g^c, g^b, g^{b^2}, ..., g^{b^\ell}, e(g,g)^z) = 1] \geq \epsilon$$

where the probability is over the random choices of $b, c, z \in \mathbb{Z}_p^*$

We say that the decisional ℓ-wBDHI* assumption holds in G if no polynomially bounded adversary has advantage better than negligible in solving the decisional ℓ-wBDHI* problem in G.

Remark 4. Cheon showed in [9] an attack against the Strong Diffie-Hellman Assumption and its related problems (among which the DBDHI used to prove the security of the Dodis-Yampolskiy VRF). This attack reduces the security of a factor $\sqrt{q}$ and applies to the ℓ-wBDHI* as well. However, as it is stated at the beginning of this section, in our construction it is enough to assume that the ℓ-wBDHI* assumption holds only for rather small values of ℓ (i.e. $\ell = 160$ or $\ell = 256$). Thus in our case the security loss is not significant as in Dodis-Yampolskiy's.

The proposed scheme follows

$\mathsf{Setup}(1^k)$ runs $\mathcal{G}(1^k)$ to obtain the description of the groups G, G_T and of a bilinear map $e : G \times G \to G_T$. The description of G contains a generator $g \in G$. Let $\{0,1\}^\ell$ be the space of valid identities. Then the algorithm picks (at random) $a, \alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, sets $g_1 = g^a$, and for $i = 1, \ldots, \ell$ sets $g_{0i} = g^{\beta_i}$ and $g_{1i} = g^{\alpha_i}$. The public parameters are

$$mpk = \left(g, g_1, \{g_{ij}\}_{i=0,1;j=1..\ell}\right)$$

The master secret key is $msk = (a, \{\alpha_i, \beta_i\}_{i=1,..,\ell})$

$\mathsf{KeyDer}(msk, ID)$ We assume $ID = ID_1 \cdots ID_\ell$ where each $ID_i \in \{0,1\}$. The key derivation algorithm constructs the secret key sk_{ID} and the auxiliary information aux_{ID} as follows. Let $h_0 = g$, for $i = 1$ to ℓ one computes

$$h_i = (h_{i-1})^{\alpha_i^{ID_i} \beta_i^{(1-ID_i)}}$$

and sets $aux_{ID} = (h_1, \ldots, h_\ell)$ and $sk_{ID} = h_\ell^a$.

$\mathsf{Encap}(mpk, ID, aux_{ID})$ Let $aux_{ID} = (h_1, \ldots, h_\ell)$ computed as above, the encapsulation algorithm picks a random $t \xleftarrow{\$} \mathbb{Z}_p^\star$ and computes a random session key $K = e(g_1, h_\ell)^t$ and a corresponding ciphertext $C = g^t$.

$\mathsf{Decap}(C, sk_{ID}, aux_{ID})$ the decapsulation algorithm uses the secret key sk_{ID} and the auxiliary information aux_{ID} to compute a session key K from a ciphertext C. This is done as follows. First, in order to guarantee the unique decryption property, a check on the validity of the auxiliary information has to be performed. This is done as follows, let $h_0 = g$, for $i = 1, \ldots, \ell$

$$\begin{aligned} \textbf{if } ID_i = 1 \textbf{ check } & e(g, h_i) \stackrel{?}{=} e(g_{1i}, h_{i-1}) \\ \textbf{else} \quad \textbf{check } & e(g, h_i) \stackrel{?}{=} e(g_{0i}, h_{i-1}) \end{aligned}$$

If any of the above checks fails output reject. Second, the key K is computed as $K = e(C, sk_{ID}) = e(g_1, h_\ell)^t$ Notice that, the validity of sk_{ID} can be verified by first encrypting some random message m with respect to the public key (g, g_1, h_ℓ) and then by checking if one can decrypt it correctly using sk_{ID}.

Now we prove the following result

Theorem 4. *Suppose the decisional ℓ-wBDHI* assumption holds in G, then the scheme given above is a secure VRF suitable IB-KEM scheme.*

Proof. Let $\mathcal{ID} = \{0,1\}^\ell$ the identity space. First notice that the scheme fits the syntax of VRF suitable IB-KEMs. We prove the theorem by showing that the scheme has the unique decryption property and meets the pseudorandom decapsulation requirement.

Unique Decryption. We prove this by showing that for a given identity ID the corresponding h_ℓ is uniquely determined as

$$h_\ell = g^{\prod_{i=1}^{\ell} \alpha_i^{ID_i} \beta_i^{1-ID_i}}$$

The proof is by induction on i. First notice that it must be the case $h_1 = g^{\alpha_1^{ID_1} \beta_1^{1-ID_1}}$, as otherwise the check $e(g, h_1) \stackrel{?}{=} e(g_{ID_1 1}, h_0) = e(g^{\alpha_1^{ID_1} \beta_1^{1-ID_1}}, g)$ would fail. Now assume that the statement holds true for any index $j - 1 < \ell$, i.e. that $h_{j-1} = g^{\prod_{i=1}^{j-1} \alpha_i^{ID_i} \beta_i^{1-ID_i}}$. We prove that the same holds for j.

$$h_j = h_{j-1}^{\alpha_j^{ID_j} \beta_j^{1-ID_j}} = \left(g^{\prod_{i=1}^{j-1} \alpha_i^{ID_i} \beta_i^{1-ID_i}} \right)^{\alpha_j^{ID_j} \beta_j^{1-ID_j}} = g^{\prod_{i=1}^{j} \alpha_i^{ID_i} \beta_i^{1-ID_i}}$$

Pseudorandom Decapsulation. Assume that there is an an adversary $\mathcal{A}$ that breaks the pseudorandom decapsulation of the proposed scheme with advantage ϵ, we show how to build an adversary $\mathcal{B}$ that solves the decisional ℓ-wBDHI* problem with advantage $\epsilon/2^\ell$ and runs in time comparable to that needed by $\mathcal{A}$. $\mathcal{B}$ starts by receiving, from some challenging oracle, the values

($C = g^c, B_1 = g^b, B_2 = g^{b^2}, \ldots B_\ell = g^{b^\ell}$ and a value T that can be either of the form $e(g,g)^{b^{\ell+1}c}$ or of the form $e(g,g)^z$, for random $z \in \mathbb{Z}_p^*$, depending on some random (and hidden) bit d that $\mathcal{B}$ is supposed to guess. First, notice that in the proposed scheme the ciphertext C is independent of specific identities, thus $\mathcal{B}$ can produce it without having to commit to any ID_0. $\mathcal{B}$ chooses $\overline{ID}$ at random as its guess for the challenge identity. Then it sets $g_1 = B_1^a$, for random $a \in \mathbb{Z}_p^*$, chooses at random $\alpha_i, \beta_i \xleftarrow{\$} \mathbb{Z}_p^*$, for $i = 1, \ldots, \ell$, and computes for $i = 1, \ldots, \ell$

$$g_{0i} = \begin{cases} B_1^{\beta_i} & if \;\; \overline{ID}_i = 0 \\ g^{\beta_i} & if \;\; \overline{ID}_i = 1 \end{cases} \qquad g_{1i} = \begin{cases} g^{\alpha_i} & if \;\; \overline{ID}_i = 0 \\ B_1^{\alpha_i} & if \;\; \overline{ID}_i = 1 \end{cases}$$

Notice that the public parameters $mpk = \left(g, g_1, \{g_{ij}\}_{i=0,1;j=1..\ell}\right)$ are distributed exactly as those produced by the setup algorithm. The master secret key is implicitly set to $msk = (ab, \{\alpha_i b^{\overline{ID}_i}, \beta_i b^{1-\overline{ID}_i}\}_{i=1,..,\ell})$. Next, $\mathcal{B}$ computes C^* as follows $C^* \leftarrow C = g^c$. Thus, C^* is also correctly distributed. Now $\mathcal{B}$ runs $\mathcal{A}$ on input (mpk, C^*, ID_0), for some randomly chosen identity ID_0. Notice that, from the received inputs, $\mathcal{A}$ gets no information at all about the $\overline{ID}$ chosen by $\mathcal{B}$, thus such a choice will be identical to the challenge identity with probability $1/2^\ell$.

Now we show how $\mathcal{B}$ can answer key derivation queries for identities $ID \neq \overline{ID}$. Since $ID \neq \overline{ID}$ there exists (at least) an index j such that $ID_j \neq \overline{ID}_j$. For such index we have that either $g_{0j} = g^{\beta_j}$ (if $ID_j = 0$) or $g_{1j} = g^{\alpha_j}$ (otherwise). This means that the h_ℓ corresponding to identity ID will contain the (unknown) b with exponent $\ell - 1$, at most. Let $n < \ell$ denote the number of positions i such that $ID_i = \overline{ID}_i$. $\mathcal{B}$ computes the h_i as follows.

$$h_1 = \begin{cases} g^{\alpha_1^{ID_1}\beta_1^{1-ID_1}} & if \;\; ID_1 \neq \overline{ID}_1 \\ B_1^{\alpha_1^{ID_1}\beta_1^{1-ID_1}} & if \;\; ID_1 = \overline{ID}_1 \end{cases}$$

$$h_2 = \begin{cases} h_1^{\alpha_2^{ID_2}\beta_2^{1-ID_2}} & if \;\; ID_2 \neq \overline{ID}_2 \\ B_1^{\alpha_2^{ID_2}\beta_2^{1-ID_2}} & if \;\; ID_2 = \overline{ID}_2 \wedge ID_1 \neq \overline{ID}_1 \\ B_2^{\alpha_2^{ID_2}\beta_2^{1-ID_2}} & if \;\; ID_2 = \overline{ID}_2 \wedge ID_1 = \overline{ID}_1 \end{cases} \quad \cdots$$

Finally, letting $\omega_{\mathcal{ID}} = \prod_{i=1}^{\ell} \alpha_i^{ID_i}\beta_i^{1-ID_i}$, h_ℓ is computed as $B_n^{\omega_{ID}}$.

The sk_{ID} is set to $B_{n+1}^{a\omega_{ID}}$. Recall that, since $n < \ell$, $\mathcal{B}$ can do this operation using the values received by the challenger. It is easy to check that both the $aux_{ID} = (h_1, \ldots, h_\ell)$ and sk_{ID} are distributed as in the real key derivation algorithm.

Once $\mathcal{A}$ is done with its first phase of key derivation queries it outputs a challenge identity ID^*. If $ID^* \neq \overline{ID}$, $\mathcal{B}$ outputs a random bit and aborts. Otherwise it constructs $K_{\overline{ID}}$ as $T^{a\omega_{\overline{ID}}}$, where $\omega_{\overline{ID}} = \prod_{i=1}^{\ell} \alpha_i^{\overline{ID}_i}\beta_i^{1-\overline{ID}_i}$ and $aux_{\overline{ID}}$ is computed as before. This time however h_ℓ is set to $B_\ell^{\omega_{\overline{ID}}}$, thus $\mathcal{B}$ will not be able to explicitly compute $sk_{\overline{ID}}$. However this is not a problem as $\mathcal{B}$ is not required to do so. Finally $\mathcal{B}$ hands $(K_{\overline{ID}}, sk_{\overline{ID}})$ to $\mathcal{A}$. $\mathcal{A}$ replies back with a guess d' ($d' = 0$

means real, $d' = 1$ means random). $\mathcal{B}$ outputs d' as well. Additional key derivation queries are dealt with as in the first phase. This completes the description of the simulator.

Now notice that if $T = e(g,g)^{b^{\ell+1}c}$, $K_{\overline{ID}}$ is a valid key for the identity $\overline{ID}$. This is because, $K_{\overline{ID}} = e(g_1, h_{\overline{ID}})^c$, where $h_{\overline{ID}}$ is the h_ℓ corresponding to identity $\overline{ID}$. Thus, $h_{\overline{ID}} = g^{b^\ell \omega_{\overline{ID}}}$

$$K_{\overline{ID}} = e(g_1, h_{\overline{ID}})^c = e(g^{ab}, g^{b^\ell \omega_{\overline{ID}}})^c = T^{a\omega_{\overline{ID}}}$$

If on the other hand T is a random value so is $K_{\overline{ID}}$. Thus, by standard calculations one gets that, if $\mathcal{A}$ has advantage ϵ in breaking the pseudorandom decapsulation property of the scheme, $\mathcal{B}$ breaks the decisional ℓ-wBDHI* with advantage $\epsilon/2^\ell$. □

Remark 5. It is interesting to note that if one is interested only into a selective-VRF, then the above construction leads directly to a scheme with large input space. This does not hold for the Dodis-Yampolskiy VRF because in its security proof the simulator has to guess all the queries that the adversary is going to ask even in the weaker selective case.

5 Conclusions

In this paper we introduced a new methodology to construct verifiable random functions (VRF) from a class of identity based key encapsulation schemes that we call VRF suitable. We showed the applicability of our methods by providing two concrete realizations of the new primitive. The first one leads to a VRF that is very similar to the Dodis-Yampolskiy construction, while the second leads to a new construction. A natural question left open by this research is to find new (potentially better) instantiations of the primitive, possibly ones supporting exponentially large (in the security parameter) identity spaces and provably secure under non interactive assumptions. This would solve the long standing open problem of realizing a secure VRF with unbounded input size.

Acknowledgements

We thank Gregory Neven for collaborating with us at an early stage of this research. We also thank Eike Kiltz and Jonathan Katz for helpful discussions. This work was supported in part by the European Commission through the IST Program under Contract ICT-2007-216646 ECRYPT II and in part by the French National Research Agency through the PACE project.

References

1. Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic constructions of identity-based and certificateless KEMs. Journal of Cryptology 21(2), 178–199 (2008)

2. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption-without pairings. In: 48th FOCS, Providence, USA, pp. 647–657. IEEE Computer Society Press, Los Alamitos (2007)
7. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
8. Chase, M., Lysyanskaya, A.: Simulatable vRFs with applications to multi-theorem NIZK. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 303–322. Springer, Heidelberg (2007)
9. Cheon, J.H.: Security analysis of the strong diffie-hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
10. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
11. Dodis, Y.: Efficient construction of (Distributed) verifiable random functions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 1–17. Springer, Heidelberg (2002)
12. Dodis, Y., Puniya, P.: Verifiable random permutations. Cryptology ePrint Archive, Report 2006/078 (2006), `http://eprint.iacr.org/`
13. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
14. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
15. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC, Seattle, Washington, USA, May 15–17, pp. 25–32. ACM Press, New York (1989)
16. Goldwasser, S., Ostrovsky, R.: Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 228–245. Springer, Heidelberg (1993)
17. Jarecki, S., Shmatikov, V.: Handcuffing big brother: an abuse-resilient transaction escrow scheme. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 590–608. Springer, Heidelberg (2004)
18. Liskov, M.: Updatable zero-knowledge databases. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 174–198. Springer, Heidelberg (2005)
19. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing 17(2) (1988)

20. Lysyanskaya, A.: Unique signatures and verifiable random functions from the DH-DDH separation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 597–612. Springer, Heidelberg (2002)
21. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: 40th FOCS, October 17–19, pp. 120–130. IEEE Computer Society Press, New York (1999)
22. Micali, S., Reyzin, L.: Soundness in the public-key model. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 542–565. Springer, Heidelberg (2001)
23. Micali, S., Rivest, R.L.: Micropayments revisited. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 149–163. Springer, Heidelberg (2002)
24. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS, Miami Beach, Florida, October 19–22, pp. 458–467. IEEE Computer Society Press, Los Alamitos (1997)
25. Sakai, R., Kasahara, M.: Id based cryptosystems with pairing on elliptic curve. In: 2003 Symposium on Cryptography and Information Security – SCIS 2003, Hamamatsu, Japan (2003), `http://eprint.iacr.org/2003/054`
26. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
27. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A The VRF by Dodis and Yampolskiy

In this section we describe the VRF by Dodis and Yampolskiy [13].

$\texttt{Gen}(1^k)$ Runs $\mathcal{G}(1^k)$ to obtain the description of the groups G, G_T and of a bilinear map $e : G \times G \to G_T$. The description of G contains a generator $g \in G$. Then the algorithm picks a random $s \xleftarrow{\$} \mathbb{Z}_p$ and sets $h = g^s$, $vpk = (g, h)$, $vsk = s$.

$\texttt{Func}_{vsk}(x)$Let $\texttt{Func}_{vsk}(x)$=$(F_{vsk}(x), \pi_{vsk}(x))$. One sets $\texttt{Func}_{vsk}(x)$ =$e(g, g)^{1/(s+x)}$ as the VRF output and $\pi_{vsk}(x) = g^{1/(s+x)}$ as the proof of correctness.

$\texttt{V}(vpk, x, y, \pi_x)$ To verify if y was computed correctly, one checks that $e(g^x \cdot h, \pi_x) = e(g, g)$ and $y = e(g, \pi_x)$.

Optimal Randomness Extraction from a Diffie-Hellman Element

Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer

École Normale Supérieure, CNRS-INRIA, Paris, France
{Celine.Chevalier,Pierre-Alain.Fouque,David.Pointcheval,
Sebastien.Zimmer}@ens.fr

Abstract. In this paper, we study a quite simple deterministic randomness extractor from random Diffie-Hellman elements defined over a prime order multiplicative subgroup G of a finite field $\mathbb{Z}_p$ (the truncation), and over a group of points of an elliptic curve (the truncation of the abscissa). Informally speaking, we show that the least significant bits of a random element in $G \subset \mathbb{Z}_p^*$ or of the abscissa of a random point in $\mathcal{E}(\mathbb{F}_p)$ are indistinguishable from a uniform bit-string. Such an operation is quite efficient, and is a good randomness extractor, since we show that it can extract nearly the same number of bits as the Leftover Hash Lemma can do for most Elliptic Curve parameters and for large subgroups of finite fields. To this aim, we develop a new technique to bound exponential sums that allows us to double the number of extracted bits compared with previous known results proposed at ICALP'06 by Fouque *et al.* It can also be used to improve previous bounds proposed by Canetti *et al.* One of the main application of this extractor is to mathematically prove an assumption proposed at Crypto '07 and used in the security proof of the Elliptic Curve Pseudo Random Generator proposed by the NIST. The second most obvious application is to perform efficient key derivation given Diffie-Hellman elements.

1 Introduction

Since Diffie and Hellman's seminal paper [10], many cryptographic schemes are based on the Diffie-Hellman technique: key exchange protocols [10] of course, but also encryption schemes, such as ElGamal [12] and Cramer-Shoup [9] ones, or pseudo-random generators, as the Naor-Reingold PRNG [23]. More precisely, the security of these schemes relies on the Decisional Diffie-Hellman assumption (DDH) [4], which means that there is no efficient algorithm that can distinguish the two distributions in G^3, (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where a, b and c are chosen at random in $[\![1, q]\!]$, and $G = \langle g \rangle$ is a cyclic group, generated by g of prime order q. For many of the schemes whose security is based on the DDH assumption, the DH element is used as a shared random element of G. However, a perfectly random element of G is not a perfectly random bit string and sometimes, as in key derivation for example, it can be useful to derive a uniform bit string which could be used as a symmetric key. Therefore, from this random element of G, one has to find a way to generate a random bit string.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 572–589, 2009.

1.1 Related Work

One classical solution to derive a random-looking bit-string from the DH element is to use a hash function. One indeed gets a uniform bit-string, but in the random oracle model [2].

Another solution, secure in the standard model, is to use a randomness extractor, such as the one which has been proposed by Gennaro *et al.* in [15]. But one first needs to have some entropy in the DH element, whereas g^{ab} is totally determined by g^a and g^b. This entropy is computationally injected using a computational assumption, as the CDH and DDH assumptions.

The CDH assumption, which states that g^{ab} is difficult to compute from g^a and g^b, implies that several bits of g^{ab} are not known from the adversary. Therefore, from the adversary point of view, there is some randomness in it. So one solution is to prove the hardness of predicting the least significant bits of a DH element. This comes from the hardcore bit theory, where one tries to provide a reduction between an algorithm that predicts the least significant bits of the DH element to the recovery of the whole DH element: predicting these bits is thus as hard as solving the CDH problem. However, usually, only a small number of bits can be proved to be random-looking, given g^a and g^b [6, 5, 20].

This entropy can also be computationally created using the DDH assumption, which says that we have $\log_2(q)$ bits of entropy in the DH element, but one does not know where exactly: one cannot extract them directly out of the representation of the element in G. This is the goal of a randomness extractor. The *Leftover Hash Lemma* [17, 15] is the most famous randomness extractor. It is a *probabilistic* randomness extractor that can extract entropy for *any* random source which has sufficient min-entropy. The main drawback with the Leftover Hash Lemma is that it requires the use of pairwise independent hash functions, which are not used in practice, and extra perfect randomness. A computational version of this Leftover Hash Lemma, has also been proposed and analysed in [14], version which has the advantage of using pseudorandom functions for randomness extraction and not pairwise independent hash functions. However, it still requires the use of extra perfect randomness. The two previous solutions are generic: it could be interesting to find a *deterministic* solution dedicated to the randomness extraction from a random element in G, since it would prevent the use of extra randomness.

Definitely, the most interesting solution in this vein is to keep the least significant bits of the DH element and hope that the resulting bit-string is uniform, as it is proposed in many papers [6, 5, 20]. Truncation, as studied above and in this paper, is quite simple and *deterministic*, which is of high interest from a practical point of view, even if it is specific to DH distributions.

A first step in this direction was the analysis of Canetti *et al.* [8] which basically shows that the concatenation of the least significant bits of g^a, g^b and g^{ab} is close to the uniform distribution. This result was achieved using exponential sums techniques. However, Boneh [4] noted: "*This is quite interesting although it does not seem to apply to the security analysis of existing protocols. In most protocols, the adversary learns all of* g^a *and* g^b*.*" This result is statistical and

no cryptographic assumption is required, since some bits of a and b are free, when the view of the adversary is limited to some part of g^a and g^b. There is no chance to extend this result to our problem, since, as already noted, given the entire representation of g^a and g^b, there is no randomness at all in g^{ab}. However, under the DDH assumption, some entropy appears in the DH element, and so, one can expect to extract it into a bit-string that will be close to the uniform distribution, in a statistical sense.

At ICALP'06, Fouque *et al.* [13] use this idea and show that under the DDH assumption, the least significant bits of g^{ab} are nearly uniformly distributed, given g^a and g^b, if the group G is a large enough multiplicative subgroup (of prime order q) of a finite field (let say $\mathbb{Z}_p$), that is, q is not too small compared to p. The large q is the main drawback since q needs to be at least half the size of p, which makes the cryptographic protocol quite inefficient. To prove this result, the authors upper bound the statistical distance, evaluating directly the L_1 norm, using exponential sums.

Since elliptic curves cryptography uses large subgroup in practice, the same result for elliptic curve could be of practical interest. Gürel [16] studied the case of elliptic curves over quadratic extensions of a finite field, with a large fraction of bits, and over a prime finite field, but with similar limitations as above in the number of extracted bits. He also upper bounds directly the statistical distance by evaluating the L_1 norm, but using a sum of Legendre characters. His technique only uses the Legendre character, which is not enough in the case of $\mathbb{Z}_p$. Consequently, the technique of the authors of [13] needed to sum on all characters.

1.2 Our Results

In this paper, we show that the following distributions are computationally indistinguishable

$$(aP, bP, U_k) \approx_C (aP, bP, \mathtt{lsb}_k(x(abP))),$$

where U_k is the uniform distribution on k-bit strings, $\mathtt{lsb_k}()$ is the function which truncates the k least significant bits of a bit-string and $x()$ is the abscissa function of points on an elliptic curve.

Under the DDH assumption, we know that $(aP, bP, abP) \approx_C (aP, bP, cP)$ for random scalars $a, b, c \in [\![1, q]\!]$, in the group G, generated by P of prime order q. Then, we prove, without any cryptographic or mathematical assumption, that

$$(aP, bP, U_k) \approx_S (aP, bP, \mathtt{lsb}_k(x(cP)))$$

in a statistical sense.

Actually, we first show this result for prime order multiplicative subgroups of finite fields. This result extends those of Canetti *et al.* and of Fouque *et al.* since we are able to extract twice the number of bits as before. This new result is achieved by introducing a new technique to bound the statistical distance. Whereas previous techniques directly tried to bound the L_1 norm, while it is hard to cope with the absolute value, we upper-bound the Euclidean L_2 norm,

which is much easier since only squares are involved. Finally, we are also able, in some cases, to improve our result using classical techniques on exponential sums. Then, the number of extracted bits can be made quite close to the number that the Leftover hash lemma can extract.

However, since the result still applies to large subgroups only, we extend it to Elliptic Curve groups. In general, the co-factor of EC groups is small: less than 8, and even equal to one for the NIST curves, over prime fields. We thus achieve our above-mentioned result using more involved techniques on exponential sums over functions defined on the points of the elliptic curve. More precisely, we can show that the 82 (resp. 214 and 346) least significant bits of the abscissa of a DH element of the NIST curves over prime fields of 256 (resp. 384 and 521) bits are indistinguishable from a random bit-string. They can thus be directly used as a symmetric key. To compare with Gürel's result in [16], for an elliptic curve defined over a prime field of 200 bits, Gürel extracts 50 bits with a statistical distance of 2^{-42}, while with the same distance, we can extract 102 bits. Note that Gürel's proof was easier to understand, but we did not manage to evaluate the L_2 norm of Legendre character sums and generalize his proof.

One main practical consequence of the result for elliptic curve is that, we can avoid the Truncated Point Problem (TPP) assumption used in the security proof of the NIST Elliptic Curve Dual Random Bit Generator (DRBG) [7, 24].

1.3 Organization of the Paper

In Section 2, we review some notations and the definition of a deterministic randomness extractor as well as some results on the Leftover Hash Lemma. Then, in Section 3, we improve the results of Canetti *et al.* and of Fouque *et al.* using a new technique to bound exponential sums, using the Euclidean norm. In this section, we also improve the bound in some cases. Next, in Section 4, we prove the same kind of result for the group of points of an elliptic curve. Finally, in Section 5, we show some applications of our proofs to the security of the NIST EC DRBG [7, 24] and the key derivation from a DH element.

2 Notations

First, we introduce the notions used in randomness extraction. In the following, a source of randomness is viewed as a probability distribution.

2.1 Measures of Randomness

To measure the randomness existing in a random variable, we use two different measures: the *min entropy* and the *collision entropy*. The min entropy measures the difficulty that an adversary has to guess the value of the random variable, whereas the collision entropy measures the probability for two elements drawn according this distrubtion to collide. In this paper, the collision entropy is used as an intermediate tool to establish results, which are then reformulated using min entropy.

Definition 1 (Min Entropy). *Let X be a random variable with values in a finite set $\mathcal{X}$. The* guessing probability *of X, denoted by $\gamma(X)$, is the probability* $\max_{x\in\mathcal{X}}(\Pr[X=x])$. *The min entropy of X is: $H_\infty(X) = -\log_2(\gamma(X))$.*

For example, when X is drawn from the uniform distribution on a set of size N, the min-entropy is $\log_2(N)$.

Definition 2 (Collision Entropy). *Let X and X' be two random independent and identically distributed variables with values in a finite set $\mathcal{X}$. The* collision probability *of X, denoted by* $\mathrm{Col}(X)$ *is the probability* $\Pr[X = X'] = \sum_{x\in\mathcal{X}} \Pr[X=x]^2$. *The* collision entropy *of X is:* $\mathbf{H}_2(X) = -\log_2(\mathrm{Col}(X))$.

The collision entropy is also called the Renyi entropy. There exists an easy relation between collision and min entropies: $\mathbf{H}_\infty(X) \leq \mathbf{H}_2(X) \leq 2\cdot\mathbf{H}_\infty(X)$. To compare two random variables we use the classical statistical distance:

Definition 3 (Statistical Distance). *Let X and Y be two random variables with values in a finite set $\mathcal{X}$. The* statistical distance *between X and Y is the value of the following expression:*

$$\mathbf{SD}(X,Y) = \frac{1}{2}\sum_{x\in\mathcal{X}} \left|\Pr[X=x] - \Pr[Y=x]\right|.$$

We denote by U_k a random variable uniformly distributed over $\{0,1\}^k$. We say that a random variable X with values in $\{0,1\}^k$ is *δ-uniform* if the statistical distance between X and U_k is upper-bounded by δ.

Lemma 4. *Let X be a random variable with values in a set $\mathcal{X}$ of size $|\mathcal{X}|$ and $\varepsilon = \mathbf{SD}(X, U_\mathcal{X})$ the statistical distance between X and $U_\mathcal{X}$, the uniformly distributed variable over $\mathcal{X}$. We have:*

$$\mathrm{Col}(X) \geq \frac{1+4\varepsilon^2}{|\mathcal{X}|}. \tag{1}$$

Proof. This lemma, whose result is very useful in this work, is proved in Appendix A.

2.2 From Min Entropy to δ-Uniformity

The most common method to obtain a δ-uniform source is to extract randomness from high-entropy bit-string sources, using a so-called *randomness extractor*. Presumably, the most famous randomness extractor is provided by the *Leftover Hash Lemma* [17, 19], which requires the use of *universal hash function families*.

Definition 5 (Universal Hash Function Family). *A universal hash function family $(h_i)_{i\in\{0,1\}^d}$ with $h_i : \{0,1\}^n \to \{0,1\}^k$, for $i \in \{0,1\}^d$, is a family of functions such that, for every $x \neq y$ in $\{0,1\}^n$, $\Pr_{i\in\{0,1\}^d}[h_i(x) = h_i(y)] \leq 1/2^k$.*

Let $(h_i)_{i\in\{0,1\}^d}$ be a universal hash function family, let i denote a random variable with uniform distribution over $\{0,1\}^d$, let U_k denote a random variable uniformly distributed in $\{0,1\}^k$, and let X denote a random variable taking values in $\{0,1\}^n$, with i and X mutually independent and with X min entropy greater than m, that is $\mathbf{H}_\infty(X) \geq m$. The Leftover Hash Lemma (which proof can be found in [25]) states that $\mathbf{SD}(\langle i, h_i(X)\rangle, \langle i, U_k\rangle) \leq 2^{(k-m)/2-1}$.

In other words, if one wants to extract entropy from the random variable X, one generates a *uniformly distributed* random variable i and computes $h_i(X)$. The Leftover Hash Lemma guarantees a 2^{-e} security, if one imposes that

$$k \leq m - 2e + 2. \tag{2}$$

The Leftover Hash Lemma extracts nearly all of the entropy available *whatever the randomness sources are*, but it needs to invest few additional truly random bits. To overcome this problem, it was proposed to use deterministic functions. They do not need extra random bits, but only exist for some specific randomness sources.

Definition 6 (Deterministic Extractor). *Let f be a function from $\{0,1\}^n$ into $\{0,1\}^k$. Let X be a random variable taking values in $\{0,1\}^n$ and let U_k denote a random variable uniformly distributed in $\{0,1\}^k$, where U_k and X are independent. We say that f is an (X,ε)-deterministic extractor if:*

$$\mathbf{SD}\left(f(X), U_k\right) < \varepsilon.$$

2.3 Characters on Abelian Groups

We recall a standard lemma for character groups of Abelian groups.

Lemma 7. *Let H be an Abelian group and $\hat{H} = \mathrm{Hom}(H, \mathbb{C}^*)$ its dual group. Then, for any element χ of $\hat{H}$, the following holds, where χ_0 is the trivial character:*

$$\frac{1}{|H|}\sum_{h\in H}\chi(h) = \begin{cases} 1 & \text{if } \chi = \chi_0 \\ 0 & \text{if } \chi \neq \chi_0 \end{cases}$$

In the following, we denote by e_p the character such that for all $x \in \mathbb{F}_p$, $e_p(x) = e^{\frac{2i\pi x}{p}} \in \mathbb{C}^*$.

2.4 Elliptic Curves

Let p be a prime and $\mathcal{E}$ be an elliptic curve over $\mathbb{F}_p$ given by the Weierstrass equation

$$y^2 + (a_1x + a_3)\cdot y = x^3 + a_2x^2 + a_4x + a_6.$$

We denote by $\mathcal{E}(\mathbb{F}_p)$ the group of elements of $\mathcal{E}$ over $\mathbb{F}_p$ and by $\mathbb{F}_p(\mathcal{E})$ the function field of the curve $\mathcal{E}$, defined as the field of fractions over the points

of $\mathcal{E}$: $\mathbb{F}_p(\mathcal{E}) = \mathbb{F}_p[X,Y]/\mathcal{E}(\mathbb{F}_p)$. It is generated by the functions x and y, satisfying the Weierstrass equation of $\mathcal{E}$, and such that $P = (x(P), y(P))$ for each $P \in \mathcal{E}(\mathbb{F}_p) \setminus \{O\}$. Let $f_a \in \mathbb{F}_p(\mathcal{E})$ be the application $f_a = a \cdot x$ where $a \in \mathbb{Z}_p^*$. If $f \in \mathbb{F}_p(\mathcal{E})$, we denote by $\deg(f)$ its degree, that is $\sum_{i=1}^{t} n_i \deg(P_i)$ if $\sum_{i=1}^{t} n_i P_i$ is the divisor of poles of f. Finally, we denote by $\Omega = \mathrm{Hom}(\mathcal{E}(\mathbb{F}_p), \mathbb{C}^*)$ the group of characters on $\mathcal{E}(\mathbb{F}_p)$, and by ω_0 the trivial character (such that $\omega_0(P) = 1$ for each P).

3 Randomness Extraction in Finite Fields

In this section, we first extends results from Fouque *et al.* [13], in order to extract bits from random elements in a multiplicative subgroup of a finite field. Then, we use the same techniques to improve the result of Canetti *et al.* [8].

3.1 Randomness Extraction

We study now the randomness extractor which consists in keeping the least significant bits of a random element from a subgroup G of $\mathbb{Z}_p^*$. The proof technique presented here allows us to extract twice the number of bits extracted by Fouque *et al.*. In the particular case when $q \geq p^{3/4}$, where q is the cardinal of G, we prove an even better result: one can extract as many bits as with the Leftover Hash Lemma. This means that, in the case when $q \geq p^{3/4}$, our extractor is as good as the Leftover Hash Lemma, but computationally more efficient and easiest to use in protocols, since it does not require extra perfect public randomness.

In the original paper, Fouque *et al.* upper bound directly the statistical distance between the extracted bits and the uniform distribution, using exponential sums. We still use them, but propose to apply exponential sum technique to upper bound the collision probability of the extracted bits. The Cauchy-Schwartz inequality allows to relate statistical distance and collision probability and to conclude. Since the distribution of extracted bits is very close to the uniform distribution, the Cauchy-Schwartz inequality is very tight. That is the reason why we do not lose much with our roundabout way. On the contrary, we are able to find a good upper bound of collision resistance, and thus the global upper bound is improved.

The result in the case when $q \geq p^{3/4}$ is elaborated on the same basic idea but requires more elaborated techniques on exponential sums to be established.

Theorem 8. *Let p be a n-bit prime, G a subgroup of $\mathbb{Z}_p^*$ of cardinal q (we denote $\ell = \log_2(q) \in \mathbb{R}$), U_G a random variable uniformly distributed in G and k a positive integer. We have:*

$$\mathbf{SD}(\mathsf{lsb}_k(U_G), U_k) \leq \begin{cases} 2^{3n/4-\ell-1} + 2^{(k-\ell)/2} & \left(\text{if } p^{3/4} \leq q\right) \\ 2^{(k+n+\log_2 n)/2-\ell} & \left(\text{if } (2^{-8}p)^{2/3} \leq q \leq p^{3/4}\right) \\ 2^{(k+n/2+\log_2 n+4)/2-5\ell/8} & \left(\text{if } p^{1/2} \leq q \leq (2^{-8}p)^{2/3}\right) \\ 2^{(k+n/4+\log_2 n+4)/2-3\ell/8} & \left(\text{if } (2^{16}p)^{1/3} \leq q \leq p^{1/2}\right). \end{cases}$$

We remind that these inequalities are non trivial only if they are smaller than 1.

Proof. We give here a sketch of proof of the theorem, the complete proofs are in the full version.

Let us define $K = 2^k$, $u_0 = \mathsf{msb}_{n-k}(p-1)$. Let denote by e_p the following character of $\mathbb{Z}_p$: for all $y \in \mathbb{Z}_p$, $e_p(y) = e^{\frac{2i\pi y}{p}} \in \mathbb{C}^*$. The character e_p is an homomorphism from $(\mathbb{Z}_p, +)$ in $(\mathbb{C}^*, \cdot)$. For all $a \in \mathbb{Z}_p^*$, let also introduce the following notation:

$$S(a, G) = \sum_{x \in G} e_p(ax).$$

The two main interests of exponential sums is that they allow to construct some characteristic functions and that in some cases we know good upper bounds for them. Thanks to these caracteristic functions one can evaluate the size of certain sets and, manipulating sums, one can upper bound the size of these sets.

In our case, we construct $\mathbb{1}(x, y, u) = \frac{1}{p} \times \sum_{a=0}^{p-1} e_p(a(g^x - g^y - Ku))$, where $\mathbb{1}(x, y, u)$ is the characteristic function which is equal to 1 if $g^x - g^y = Ku \bmod p$ and 0 otherwise. Therefore, we can evaluate $\mathrm{Col}(\mathsf{lsb}_k(U_G))$ where U_G is uniformly distributed in G:

$$\begin{aligned}\mathrm{Col}(\mathsf{lsb}_k(U_G)) &= \frac{1}{q^2} \times \left|\{(x, y) \in [\![0,\, q-1]\!]^2 \mid \exists u \le u_0,\, g^x - g^y = Ku \bmod p\}\right| \\ &= \frac{1}{q^2 p} \times \sum_{x=0}^{q-1} \sum_{y=0}^{q-1} \sum_{u=0}^{u_0} \sum_{a=0}^{p-1} e_p(a(g^x - g^y - Ku)).\end{aligned}$$

Then we manipulate the sums, separate some terms ($a = 0$) and obtain:

$$\mathrm{Col}(\mathsf{lsb}_k(U_G)) = \frac{u_0 + 1}{p} + \frac{1}{q^2 p} \sum_{a=1}^{p-1} |S(a, G)|^2 \left(\sum_{u=0}^{u_0} e_p(-aKu) \right). \tag{3}$$

The last three bounds. From this point, the proof of the last three inequations is different from the proof of the first inequation. First, we give here the sketch of proof of the last three inequations of the theorem (we remind that the complete proof is given in the full version).

In Equation (3) we inject the absolute value, introduce $M = \max_a(|S(a, G)|)$, make classical manipulations and obtain:

$$\mathrm{Col}(\mathsf{lsb}_k(U_G)) \le \frac{u_0 + 1}{p} + \frac{M^2 \log_2(p)}{q^2}.$$

We now use the Lemma 4 which gives a relation between the statistical distance ε of $\mathsf{lsb}_k(X)$ with the uniform distribution and the collision probability: $\mathrm{Col}(\mathsf{lsb}_k(U_G)) \ge \frac{1+4\varepsilon^2}{2^k}$. The previous upper bound, combined with some manipulations, gives:

$$2\varepsilon \le \sqrt{2^k \cdot \mathrm{Col}(\mathsf{lsb}_k(U_G)) - 1} \le \sqrt{\frac{2^k}{p}} + \frac{2^{k/2} M \log_2^{1/2}(p)}{q}. \tag{4}$$

We conclude the theorem using the following upper bounds for M:

$$M \leq \begin{cases} p^{1/2} & \left(\text{interesting if } p^{2/3} \leq q\right) \\ 4p^{1/4}q^{3/8} & \left(\text{interesting if } p^{1/2} \leq q \leq p^{2/3}\right) \\ 4p^{1/8}q^{5/8} & \left(\text{interesting if } 2^{16/3}p^{1/3} \leq q \leq p^{1/2}\right). \end{cases}$$

The first bound above is the famous Polya-Vinogradov bound that we recall in Theorem 9 (its proof is reminded in the full version). The other bounds are from [22, 18]. The last third bounds of the theorem can be easily deduced.

Theorem 9 (Polya-Vinogradov inequality). *Let p be a prime number, G a subgroup of $(\mathbb{Z}_p^*, \cdot)$. For all $a \in \mathbb{Z}_p^*$, we have:*

$$\left| \sum_{x \in G} e_p(ax) \right| \leq \sqrt{p}.$$

The first bound. We give now a sketch of proof of the first inequality, a precise proof is given in the full version. For that, we use a bit more elaborated results than previously: for all coset $\omega \in \mathbb{Z}_p^*/G$, and for all two representatives a and a' of the coset ω, we have $S(a, G) = S(a', G)$. Therefore we can naturally define $S(\omega, G)$.

To establish the first inequality, we use Equation (3) and manipulating sums we establish that:

$$\mathrm{Col}(\mathsf{lsb}_k(U_G)) = \frac{u_0 + 1}{p} + \frac{1}{q^2 p} \sum_{\omega \in \mathbb{Z}_p^*/G} |S(\omega, G)|^2 \sum_{u=0}^{u_0} S(-\omega K u, G).$$

Then we use the Polya-Vinogradov inequality combined with the inequality $\sum_{\omega \in \mathbb{Z}_p^*/G} |S(\omega, G)|^2 \leq p$ (the proof of this result is reminded in the full version) and show that:

$$\mathrm{Col}(\mathsf{lsb}_k(U_G)) \leq \frac{u_0 + 1}{p} + \frac{u_0\sqrt{p} + q}{q^2}.$$

Finally, we finish, as for previous inequalities, using that $\mathrm{Col}(\mathsf{lsb}_k(U_G)) \geq \frac{1+4\varepsilon^2}{2^k}$, and obtain:

$$2\varepsilon \leq 2^{(k-n+1)/2} + 2^{3n/4-\ell} + 2^{(k-\ell)/2}.$$

Since $\ell \leq n - 1$, this gives the expected bound. □

Since the min entropy of U_G, as an element of $\mathbb{Z}_p^*$ but uniformly distributed in G, equals $\ell = \log_2(|G|) = \log_2(q)$, the previous proposition leads to:

Corollary 10. *Let e be a positive integer and let us suppose that one of these inequations is true:*

$$k \leq \begin{cases} \ell - (2e + 2) & \text{and } 2^e \cdot p^{3/4} \leq q \\ 2\ell - (n + 2e + \log_2(n)) & \text{and } (2^{-8} \cdot p)^{2/3} \leq q \leq 2^e \cdot p^{3/4} \\ 5\ell/4 - (n/2 + 2e + \log_2(n) + 4) & \text{and } p^{1/2} \leq q \leq (2^{-8} \cdot p)^{2/3} \\ 3\ell/4 - (n/4 + 2e + \log_2(n) + 4) & \text{and } (2^{16} \cdot p)^{1/3} \leq q \leq p^{1/2}. \end{cases}$$

In this case, the application Ext_k is an $(U_G, 2^{-e})$-deterministic extractor.

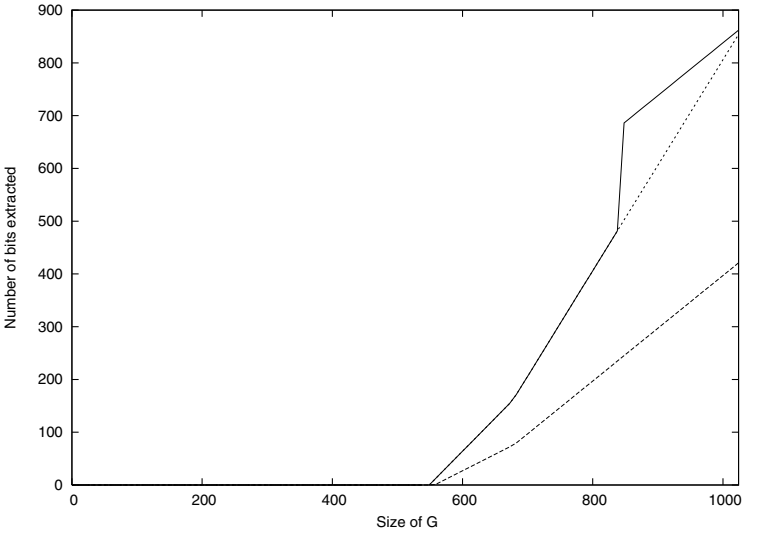


Fig. 1. This is the number of bits extracted according to the group size, for $n = 1024$ and $e = 80$. The long dash line represents Fouque *et al.* [13] result, the plain line is our results. Note the jump for $q = p^{3/4}$. The short dash line represents our result without particular improvement in the case $q \geq p^{3/4}$.

This means that if one wants a 2^{-e} security, and if $(2^{-8} \cdot p)^{2/3} \leq q \leq 2^e \cdot p^{3/4}$, one can extract k bits with $k \leq 2(\ell - (n/2 + e + \log_2 n/2))$.

In most practical cases, the second bound is the most appropriate. However, sometimes it is one of the others. For example, with $n = 1024$, $\ell = 600$ and $e = 80$, the second bound says that we can extract 6 bits. Using the third bound given in the theorem above we can actually extract 64 bits.

If one wants to extract a 256-bit string, for the same values of n and e, one needs a group of size greater than 2^{756}. The figure 1 presents our upper bounds and also the original upper bounds of Fouque *et al.* [13], in the case when $n = 1024$ and $e = 80$.

3.2 Truncated Inputs

Our above result proves that given g^a and g^b, the least significant bits of g^{ab} are globally indistinguishable from a random bit-string, under the Decisional Diffie-Hellman problem.

But our technique can be applied to other results which upper-bound statistical distances using character sums. One of them is the result of Canetti *et al.* [8], which studies some statistical properties of Diffie-Hellman distribution. They show that if one takes a proportion of the least significant bits of g^x, g^y, g^{xy}, then one obtains a distribution whose statistical distance from uniform is exponentially small. Basically, it shows that given the least significant bits of

g^a and g^b, the least significant bits of g^{ab} are globally indistinguishable from a random bit-string, without any computational assumption.

More precisely, if k_1, k_2, k_3 are three integers and U_1, U_2, U_3 three independent random variables uniformly distributed in respectively $\{0,1\}^{k_1}$, $\{0,1\}^{k_2}$, $\{0,1\}^{k_3}$, then, using the notations as in previous subsection, their Theorem 9 inequality, can be restated as follows:

$$\mathbf{SD}\left(\left(\mathsf{lsb}_{k_1}\left(g^x\right), \mathsf{lsb}_{k_2}\left(g^y\right), \mathsf{lsb}_{k_3}\left(g^{xy}\right)\right), (U_1, U_2, U_3)\right) \ll \frac{2^{k_1+k_2+k_3} p^{1/4} \log_2^{1/3}(p)}{q^{1/3}}.$$

Using our techniques, we can prove a better upper-bound:

Theorem 11. *Let p be a prime, G a subgroup of $\mathbb{Z}_p^*$ of cardinal q and X, Y two independent random variables uniformly distributed in $\{1, \ldots, q\}$. If k_1, k_2, k_3 are three integers and U_1, U_2, U_3 three independent random variables uniformly distributed in respectively $\{0,1\}^{k_1}$, $\{0,1\}^{k_2}$, $\{0,1\}^{k_3}$, then we have:*

$$\mathbf{SD}\left(\left(\mathsf{lsb}_{k_1}\left(g^x\right), \mathsf{lsb}_{k_2}\left(g^y\right), \mathsf{lsb}_{k_3}\left(g^{xy}\right)\right), (U_1, U_2, U_3)\right) \ll \frac{2^{\frac{k_1+k_2+k_3}{2}} p^{1/4} \log_2^{1/3}(p)}{q^{1/3}}.$$

Proof (Sketch of the proof.). First, find an upper bound for the collision entropy $\mathrm{Col}\left(\mathsf{lsb}_{k_1}\left(g^x\right), \mathsf{lsb}_{k_2}\left(g^y\right), \mathsf{lsb}_{k_3}\left(g^{xy}\right)\right)$ using exponential sum techniques and the inequality of Theorem 8 of [8]. Conclude with Lemma 4. □

4 Randomness Extraction in Elliptic Curves

We now show how the randomness extractor studied in the previous section, which consisted in keeping the least significant bits of a random element from a subgroup G of $\mathbb{Z}_p^*$, can be extended to another structure, that is the group of elements of an elliptic curve. The main idea is to evaluate the element "M" of the proof of Theorem 8, which is the upper bound of $S(a, G)$ as defined in the previous section.

4.1 A bound for $S(a, G)$

If G is a subgroup of $\mathcal{E}(\mathbb{F}_p)$, $f \in \mathbb{F}_p(\mathcal{E})$ and $\omega \in \Omega$, we define

$$S(\omega, f, G) = \sum_{P \in G} \omega(P) e_p(f(P)).$$

In particular, since $e_p \in \Omega$,

$$S(a, G) = S(\omega_0, f_a, G) = \sum_{P \in G} e_p(f_a(P)).$$

The objective of this section is to show the following result:

Theorem 12. *Let $\mathcal{E}$ be an elliptic curve over $\mathbb{F}_p$ and $f \in \mathbb{F}_p(\mathcal{E})$. Then,*

$$S(\omega, f, \mathcal{E}(\mathbb{F}_p)) \leq 2 \deg(f) \sqrt{p}.$$

As a consequence, if $a \in \mathbb{Z}^*$, $S(a, \mathcal{E}(\mathbb{F}_p)) \leq 4\sqrt{p}$.

More specifically, in the case where the co-factor is not equal to 1, we are interested in its corollary. Note that in the case of $\mathbb{F}_p$, all the curves recommended by the NIST have co-factor equal to 1. The proof of this corollary can be found in the full version.

Corollary 13. *Let $\mathcal{E}$ be an elliptic curve over $\mathbb{F}_p$, $a \in \mathbb{Z}^*$ and G a subgroup of $\mathcal{E}(\mathbb{F}_p)$. Then,*

$$S(\omega, f, G) \leq 2\deg(f)\sqrt{p} \qquad \text{and} \qquad S(a, G) \leq 4\sqrt{p}.$$

Proof (of Theorem 12). For sake of simplicity, we only show the case where $\omega = \omega_0$ in order to use easier notations. We follow the proof of Bombieri in [3] and Kohel and Shparlinski in [21], by first considering $S_m(f, \mathcal{E}(\mathbb{F}_p)) = S(\sigma \circ f, \mathcal{E}(\mathbb{F}_{p^m}))$ where σ is the trace from $\mathbb{F}_{p^m}$ to $\mathbb{F}_p$. Note that for our needs, the interesting sum corresponds to $m = 1$.

This sum comes from the character $e_p \circ f$, which defines an Artin-Schreier extension (informally, an extension of degree p) of the function field $\mathbb{F}_p(\mathcal{E})$, and then an Artin-Schreier covering of $\mathcal{E}(\mathbb{F}_p)$. An easy way to evaluate this sum is to consider the L-function related to this Artin-Schreier covering. L-functions are a standard means to assemble several elements in a unique object (a series), in the same manner as a generating power series, see for example [26, chap. 14]. Bombieri shows that this L-function is defined as follows, for $t \in \mathbb{C}$ such that $|t| < q^{-1}$:

$$L(t, f, \mathcal{E}(\mathbb{F}_p)) = \exp\left(\sum_{m=1}^{+\infty} S(f, \mathcal{E}(\mathbb{F}_p))t^m/m\right).$$

By the Artin conjecture, which proof was given by Weil in [27] (see the full version), this function is a polynomial of degree $D = \deg(f)$. Denote its D complex roots (not necessarily distincts) by $\theta_i = {\omega_i}^{-1}$. Then, we have the two following equations:

$$L(t, f, \mathcal{E}(\mathbb{F}_p)) = \sum_{i=0}^{+\infty} \frac{1}{i!}\left(\sum_{m=1}^{+\infty} S_m(f, \mathcal{E}(\mathbb{F}_p))t^m/m\right)^i$$

$$L(t, f, \mathcal{E}(\mathbb{F}_p)) = \prod_{i=1}^{D}(1 - \omega_i t).$$

The first equation can be rewritten the following way:

$$1 + \sum_{m=1}^{+\infty} S_m(f, \mathcal{E}(\mathbb{F}_p))t^m/m + \frac{1}{2}\sum_{m=1}^{+\infty}\sum_{n=1}^{+\infty} \frac{S_m(f, \mathcal{E}(\mathbb{F}_p))S_n(f, \mathcal{E}(\mathbb{F}_p))}{m\ n} t^{m+n} + \ldots$$

If we consider the coefficient of the polynomial of order 1, we obtain:

$$S_1(f, \mathcal{E}(\mathbb{F}_p)) = -\sum_{i=1}^{D} \omega_i.$$

The Riemann hypothetis for function fields (see [27] for the proof and the full version for the statement) shows that each zero of the above L-function verifies $|\theta_i| = 1/\sqrt{p}$. This boils down to $|S_1(f, \mathcal{E}(\mathbb{F}_p))| \leq \deg(f)\sqrt{p}$, which is the result required. Finally, we conclude by remarking that $\deg(f_a) = 2$. □

4.2 Randomness Extraction

We now show an equivalent of Theorem 8:

Theorem 14. *Let p be a n-bit prime, G a subgroup of $\mathcal{E}(\mathbb{F}_p)$ of cardinal q generated by P_0, q being a ℓ-bit prime, U_G a random variable uniformly distributed in G and k a positive integer. We have:*

$$\mathbf{SD}(\mathsf{lsb}_k(U_G), U_k) \leq 2^{(k+n+\log_2 n)/2+3-\ell}.$$

Proof. We follow the proof of Theorem 8, by constructing $\mathbb{1}(r,s,u) = \frac{1}{p} \times \sum_{a=0}^{p-1} e_p(a(f(rP_0) - f(sP_0) - Ku))$, where $\mathbb{1}(r,s,u)$ is the characteristic function which is equal to 1 if $f(rP_0) - f(sP_0) = Ku \bmod p$ and 0 otherwise. Therefore, we can evaluate $\mathrm{Col}(\mathsf{lsb}_k(x(U_G)))$ where U_G is uniformly distributed in G, exactly in the same way, and inject $M \leq 4\sqrt{p}$ in Equation (4) to obtain:

$$2\varepsilon \leq \sqrt{\frac{2^k}{p}} + \frac{2^{k/2+2}\sqrt{p}\sqrt{\log_2(p)}}{q}.$$

We conclude as before using the two inequalities $2^{n-1} < p \leq 2^n$ and $2^{\ell-1} < q \leq 2^\ell$ and remarking that the first term is negligible with respect to the second one:

$$2\varepsilon \leq 2^{(k-n-1)/2} + 2^{(k+n+\log_2(n))/2+3-\ell}. \qquad \square$$

Using the co-factor $\alpha = |\mathcal{E}(\mathbb{F}_p)| / |G| \leq 2^{n-\ell}$ of the elliptic curve, we obtain the following result:

Corollary 15. *Let e be a positive integer and let us suppose that this inequation is true:*

$$k \leq 2\ell - (n + 2e + \log_2(n) + 6) = n - (2\log_2(\alpha) + 2e + \log_2(n) + 6).$$

In this case, the application Ext_k is an $(U_G, 2^{-e})$-deterministic extractor.

5 Applications

Our extractor can be applied in every protocol which generates (possibly under the DDH assumption) a uniformly distributed element in a subgroup of $\mathbb{Z}_p^*$ or a random point over an elliptic curve, while a random bit-string is required afterwards. Our results are indeed quite useful in cryptographic protocols and primitives where one has to extract entropy from a Diffie-Hellman element.

5.1 Key Extraction

The most well known cryptographic primitive where randomness extractors are required is the key extraction phase of a key exchange protocol in order to create a secure channel. The key exchange can be either interactive (classical 2-party or group key exchange) or non-interactive (the Key Encapsulation Mechanism of an

hybrid encryption scheme). After a Diffie-Hellman key exchange (or ElGamal-like key encapsulation) performed over a group G, the parties share a common Diffie-Hellman element, which is indistinguishable from a uniformly distributed element in G granted the DDH assumption. However, they need a uniformly distributed bit-string to key a symmetric primitive: one thus extracts entropy from the DH element using a randomness extractor.

The two most well-known tools used for this task are hash functions (seen as random oracles [2]) and universal hash functions (in order to apply the Leftover Hash Lemma). Hash functions are the most often adopted solution, because of their flexibility and efficiency. However, they have a significant drawback: the validity of this technique holds in the random oracle model only. On the contrary, the Leftover Hash Lemma shows that the use of universal hash functions is secure in the standard model and that, if the cardinal of the group G is equal to q and if one wants to achieve a security of 2^{-e}, then one can extract $k = \log_2 q - 2e$ bits. However this solution requires some extra, public and perfectly random bits, which increases both time and communication complexities of the underlying protocols.

The truncation of the bit-string representation of the random element is definitely the most efficient randomness extractor, since it is deterministic, and it does not require any computation. However, the original results presented in [13, 16] were not as good as the Leftover Hash Lemma, from the number of extracted bit point of view. One could extract much less than $\log_2 q - 2e$ bits. In this paper, for large subgroups of $\mathbb{Z}_p^*$ (when the order q is larger than $p^{3/4} \cdot 2^e$), one extracts up to $\log_2 q - 2e$ bits, which is as good as the Leftover Hash Lemma. For large subgroups of an elliptic curve over $\mathbb{F}_p$, one extracts $n - 2e - \log_2(n) - 2\log_2(\alpha) - 6$ bits where α is the co-factor of the elliptic curve, which is not far from the Leftover Hash Lemma since, in practice, α is very small (often equal to 1). And then, for usual finite field size (p between 256 and 512), one can extract approximately $n - 2e - 16$.

Even with our improvement, the simple extractor may seem not very practical for subgroups of $\mathbb{Z}_p^*$, since quite large subgroups are needed. Indeed to generate a 256-bit string, with a 80-bit security and a 1024-bit prime p, one requires a 725-bit order subgroup, when the Leftover Hash Lemma would need a 416-bit order subgroup only: the time for exponentiation is approximately doubled. Note that, however, one saves on the time of the generation of extra randomness. Anyway, on elliptic curves, the improvement is quite meaningful, since groups in use are already quite large. The NIST elliptic curves have co-factor 1, and then on the 256-bit finite field elliptic curve, one can extract 82 bits, with a 80-bit security. On the 384-bit finite field, 214 bits can be extracted, while we can get 346 bits on the 521-bit field. This is clearly enough as symmetric key material for both privacy and authentication, without any additional cost.

We insist on the fact that it can apply for interactive key exchange, but also for the ElGamal [11] or Cramer-Shoup [9] encryption schemes.

5.2 NIST Random Generator

The very recent NIST SP 800-90 elliptic curve Dual Random Bit Generator (DRBG) [24] has been approved in the ANSI X9.82 standard in 2006. Based on

the elliptic curves, the design of this random bit generator (RBG) is adapted from the Blum-Micali generator. At Crypto '07, Brown and Gjøsteen [7] adapted the Blum-Micali RNG security proof to show that the DRBG output is indistinguishable from a uniformly distributed bit-string for all computationally limited adversaries. In this section, we show that our result allows to improve this security result at two different places. The first improvement reduces the number of assumptions on which the security proof relies. The second one decreases the implicit security bound given in [7].

Getting Rid of TPP Assumption. The security result of [7] holds under three computational assumptions: the classical decisional Diffie-Hellman problem (DDH), the new x-logarithm problem (XLP) (which states that, given an elliptic curve point, it is hard to determine whether the discrete logarithm of this point is congruent to the x-coordinate of an elliptic curve point), and the truncated point problem (TPP). The latter TPP states that, given a k-bit string, it is hard to tell if it was generated by truncating the x-coordinate of a random elliptic curve point or if it was chosen uniformly at random. This problem is exactly the problem we studied in this paper. In section 4, we proved that this problem is indeed hard if the elliptic curve is defined over $\mathbb{F}_p$ (where p is an n-bit prime) and if $k = n - 2e - 2\log_2(\alpha) - \log_2(n)$ bits are kept after the truncation (we remind that e denotes the expected security level and α the cofactor of the elliptic curve). Therefore, our result strengthens the security proof of [7] since thanks to it, when the elliptic curve is defined over $\mathbb{F}_p$ of appropriate size, the TPP assumption actually holds, and thus their security proof relies on the DDH and XLP assumptions only.

It is interesting to note that Brown and Gjøsteen [7], when making their highly heuristic assumptions, estimated that the expected number of bits that could be kept after truncation would be approximately $k = n - 2e - C$ where C is some constant (if the cofactor of the elliptic curve is equal to 1). Our result confirms this heuristic analysis, but is more precise since it *proves* that *in all cases* we can keep *at least* $k = n - 2e - \log_2(n)$ bits. However, we recall Brown and Gjøsteen's warning and recommend to skip $2e + \log_2(n)$ bits of the elliptic curve point abscissa in the ECRNG.

Improvement of the Security Bound. Finally, our result also allows to improve the security bound of [7]. For the sake of clarity, this security bound is not explicitly stated in [7], but can be recovered from the proof. At the very last stage of the proof, the TPP assumption is used to show that if $Z_1, \ldots, Z_m$ are uniformly distributed points on the elliptic curve and if $b_1, \ldots, b_m$ are uniformly distributed k-bit strings, then $(\mathsf{lsb}_k(Z_1), \ldots, \mathsf{lsb}_k(Z_m))$ is indistinguishable from $(b_1, \ldots, b_m)$. If any adversary has a probability of successfully distinguishing $\mathsf{lsb}_k(Z_1)$ from b_1 smaller than δ, a classical hybrid argument implies that any adversary has a probability of successfully distinguishing $(\mathsf{lsb}_k(Z_1), \ldots, \mathsf{lsb}_k(Z_m))$ from $(b_1, \ldots, b_m)$ smaller than $m \cdot \delta$. This bound can be improved to $\sqrt{2m/\pi} \cdot \delta$.

First, notice that in our case, δ is equal to $2^{(k+\log_2 n+2\log_2(\alpha)-n)/2}$. Using a result that can be found in [1], one can show that the advantage of the best adversary in distinguishing to two above m-uples is approximately equal to $\sqrt{m \cdot (2^k \cdot \mathrm{Col}(\mathsf{lsb}_k(Z_1)) - 1)/2\pi}$, if $2^k \cdot \mathrm{Col}(\mathsf{lsb}_k(Z_1)) - 1 \ll 1$. The latter expression $2^k \cdot \mathrm{Col}(\mathsf{lsb}_k(Z_1)) - 1$ is exactly the one we upper-bounded in the proof in Section 4: it is smaller than $2^{k+\log_2(n)+2\log_2(\alpha)-n+2} = 4\delta^2$. This implies that, if $\delta \ll 1$, the advantage of the best adversary in distinguishing $(\mathsf{lsb}_k(Z_1), \ldots, \mathsf{lsb}_k(Z_m))$ from $(b_1, \ldots, b_m)$ is upper bounded by $\sqrt{2m/\pi} \cdot \delta$. We thus improve the bound from [7] by a factor $\sqrt{m}$.

Acknowledgements

This work was supported in part by the French ANR-07-SESU-008-01 PAMPA Project and the European ICT-2007-216646 ECRYPT II Contract.

References

1. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
3. Bombieri, E.: On exponential sums in finite fields. American Journal of Mathematics 88, 71–105 (1966)
4. Boneh, D.: The decision diffie-hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
5. Boneh, D., Shparlinski, I.E.: On the unpredictability of bits of the elliptic curve diffie–hellman scheme. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 201–212. Springer, Heidelberg (2001)
6. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (1996)
7. Brown, D.R.L., Gjøsteen, K.: A security analysis of the NIST SP 800-90 elliptic curve random number generator. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 466–481. Springer, Heidelberg (2007)
8. Canetti, R., Friedlander, J., Konyagin, S., Larsen, M., Lieman, D., Shparlinski, I.: On the Statistical Properties of Diffie-Hellman Distributions. Israel Journal of Mathematics 120, 23–46 (2000)
9. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
10. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
11. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)

12. El Gamal, T.: On computing logarithms over finite fields. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 396–402. Springer, Heidelberg (1986)
13. Fouque, P.-A., Pointcheval, D., Stern, J., Zimmer, S.: Hardness of distinguishing the MSB or LSB of secret keys in diffie-hellman schemes. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 240–251. Springer, Heidelberg (2006)
14. Fouque, P.-A., Pointcheval, D., Zimmer, S.: HMAC is a randomness extractor and applications to TLS. In: Abe, M., Gligor, V.D. (eds.) ASIACCS, pp. 21–32. ACM Press, New York (2008)
15. Gennaro, R., Krawczyk, H., Rabin, T.: Secure hashed diffie-hellman over non-DDH groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 361–381. Springer, Heidelberg (2004)
16. Gürel, N.: Extracting bits from coordinates of a point of an elliptic curve. Cryptology ePrint Archive, Report 2005/324 (2005), `http://eprint.iacr.org/`
17. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
18. Heath-Brown, D.R., Konyagin, S.: New bounds for Gauss sums derived from k^{th} powers, and for Heilbronn's exponential sum. Q. J. Math. 51(2), 221–235 (2000)
19. Impagliazzo, R., Zuckerman, D.: How to recycle random bits. In: Proc. of the 30th FOCS, pp. 248–253. IEEE, New York (1989)
20. Jetchev, D., Venkatesan, R.: Bits security of the elliptic curve diffie–hellman secret keys. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 75–92. Springer, Heidelberg (2008)
21. Kohel, D.R., Shparlinski, I.E.: On exponential sums and group generators for elliptic curves over finite fields. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 395–404. Springer, Heidelberg (2000)
22. Konyagin, S.V., Shparlinski, I.: Character Sums With Exponential Functions and Their Applications. Cambridge University Press, Cambridge (1999)
23. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS, pp. 458–467. IEEE Computer Society Press, Los Alamitos (1997)
24. NIST. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST Special Publications 800-90 (March 2007), `http://csrc.nist.gov/publications/PubsSPs.html`
25. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, Cambridge (2005)
26. Washington, L.: Elliptic Curves: Number Theory and Cryptography. CRC Press, Boca Raton (2003)
27. Weil, A.: Sur les courbes algébriques et les variétés qui s'en déduisent. In: Actualités scientifiques et industrielles, Publications de l'institut de Mathématique de l'université de Strasbourg, vol. 1041, Paris, Hermann (1948)

A Relation between Collision Probability and Statistical Distance

In this section we prove the following lemma. The proof is taken from [25] and is given here for the sake of completeness. Note that this lemma is a consequence of the Cauchy-Schwarz inequality, which implies that the smaller the statistical distance is, the tighter the inequation is (if X is uniformly distributed, then the inequality is an equality).

Lemma 4. Let X be a random variable with values in a set $\mathcal{X}$ of size $|\mathcal{X}|$ and $\varepsilon = SD(X, U_{\mathcal{X}})$ be the statistical distance between X and $U_{\mathcal{X}}$ a random variable uniformly distributed over $\mathcal{X}$. We have:

$$\mathrm{Col}(X) \geq \frac{1 + 4\varepsilon^2}{|\mathcal{X}|}.$$

To prove the lemma we need the following result which states that norm 1 is smaller than norm 2.

Lemma 16. *Let $\mathcal{X}$ be a finite set and $(\alpha_x)_{x\in\mathcal{X}}$ a sequence of real numbers. We have:*

$$\frac{\left(\sum_{x\in\mathcal{X}} |\alpha_x|\right)^2}{|\mathcal{X}|} \leq \sum_{x\in\mathcal{X}} \alpha_x^2. \tag{5}$$

Proof. This inequality is a direct consequence of Cauchy-Schwarz inequality:

$$\sum_{x\in\mathcal{X}} |\alpha_x| = \sum_{x\in\mathcal{X}} |\alpha_x| \cdot 1 \leq \sqrt{\sum_{x\in\mathcal{X}} \alpha_x^2} \cdot \sqrt{\sum_{x\in\mathcal{X}} 1^2} \leq \sqrt{|\mathcal{X}|}\sqrt{\sum_{x\in\mathcal{X}} \alpha_x^2}.$$

The result can be deduced easily. □

If X is a random variable with values in $\mathcal{X}$ and if we consider that $\alpha_x = \Pr[X = x]$, then, since the sum of probabilities is equal to 1, and since $\mathrm{Col}(X) = \sum_{x\in\mathcal{X}} \Pr[X = x]^2$, we have:

$$\frac{1}{|\mathcal{X}|} \leq \mathrm{Col}(X). \tag{6}$$

We are now able to prove the above Lemma 4.

Proof. If $\varepsilon = 0$ the result is an easy consequence of equation Equation (6). Let assume that ε is different from 0. Let define $q_x = |\Pr[X = x] - 1/|\mathcal{X}||/(2\varepsilon)$, we have $\sum_x q_x = 1$. According to equation Equation (5), we have:

$$\begin{aligned}\frac{1}{|\mathcal{X}|} \leq \sum_{x\in\mathcal{X}} q_x^2 &= \sum_{x\in\mathcal{X}} \frac{\left(\Pr[X = x] - 1/|\mathcal{X}|\right)^2}{4\varepsilon^2} = \frac{1}{4\varepsilon^2}\left(\sum_{x\in\mathcal{X}} \Pr[X = x]^2 - 1/|\mathcal{X}|\right) \\ &\leq \frac{1}{4\varepsilon^2}\left(\mathrm{Col}(X) - 1/|\mathcal{X}|\right).\end{aligned}$$

The lemma can be deduced easily. □

A New Randomness Extraction Paradigm for Hybrid Encryption

Eike Kiltz[1], Krzysztof Pietrzak[1], Martijn Stam[2], and Moti Yung[3]

[1] Cryptology & Information Security Group
CWI Amsterdam, The Netherlands
{kiltz,pietrzak}@cwi.nl
[2] LACAL, EPFL, Switzerland
martijn.stam@epfl.ch
[3] Google Inc. and Columbia University, USA
moti@cs.columbia.edu

Abstract. We present a new approach to the design of IND-CCA2 secure hybrid encryption schemes in the standard model. Our approach provides an efficient generic transformation from 1-universal to 2-universal hash proof systems. The transformation involves a randomness extractor based on a 4-wise independent hash function as the key derivation function. Our methodology can be instantiated with efficient schemes based on standard intractability assumptions such as Decisional Diffie-Hellman, Quadratic Residuosity, and Paillier's Decisional Composite Residuosity. Interestingly, our framework also allows to prove IND-CCA2 security of a hybrid version of 1991's Damgård's ElGamal public-key encryption scheme under the DDH assumption.

Keywords: Chosen-ciphertext security, hybrid encryption, randomness extraction, hash proof systems, ElGamal

1 Introduction

Chosen-Ciphertext Security. Indistinguishability against chosen-ciphertext attack (IND-CCA2 security) is by now the accepted standard security definition for public-key encryption schemes. It started with the development of security under lunchtime attacks (also called IND-CCA1) by Naor and Yung [20], who also gave a proof of feasibility using inefficient non-interactive zero-knowledge techniques. This was extended to the more involved systems with IND-CCA2 security in their full generality [22,9].

Known practical constructions. Efficient designs in the standard model were first presented in the breakthrough works of Cramer and Shoup [2,3,4,24]. At the heart of their design methodology is the notion of *hash proof systems* (HPSs), generalizing the initial system based on the decisional Diffie-Hellman (DDH) problem. Moreover, they are the first to formalize the notion of "Hybrid Encryption," where a public key cryptosystem is used to encapsulate the (session) key of a symmetric cipher which is subsequently used to conceal the data.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 590–609, 2009.

This is also known as the KEM-DEM approach, after its two constituent parts (the KEM for key encapsulation mechanism, the DEM for data encapsulation mechanism); it is the most efficient way to employ a public key cryptosystem (and encrypting general strings rather than group elements).

Kurosawa and Desmedt [17] later improved upon the original work of Cramer and Shoup with a new paradigm. Whereas Cramer and Shoup [4] require both the KEM and the DEM IND-CCA2 secure, Kurosawa and Desmedt show that with a stronger requirement on the DEM (i.e., one-time authenticated encryption), the requirement on the KEM becomes weaker and can be satisfied with any strongly 2-universal hash proof system. (Cramer and Shoup need both a 2-universal and a smooth hash proof system.)

Main Result. The main result of this work is a new paradigm for constructing IND-CCA2 secure hybrid encryption schemes, based on the Kurosawa-Desmedt paradigm. At its core is a surprisingly clean and efficient new method employing *randomness extraction* (as part of the key derivation) to transform a universal$_1$ hash proof system (that only assures IND-CCA1 security) into a universal$_2$ hash proof system. In fact, our method also works for a more general class of hash proof systems which we denote "κ-entropic" hash proof systems. From that point on we follow the Kurosawa-Desmedt paradigm: the combination of a universal$_2$ HPS with a one-time authenticated encryption scheme (as DEM) will provide an IND-CCA2 secure hybrid encryption scheme. The efficient transformation enables the design of new and efficient IND-CCA2 secure hybrid encryption schemes based on various hard subset membership problem, such as the DDH assumption, Paillier's Decisional Composite Residuosity (DCR) assumption [21], the family of Linear assumptions [14,23] that generalizes DDH, and the Quadratic Residuosity (QR) assumption.

For the new transformation to work we require a sufficiently compressing 4-wise independent hash function (made part of the public key); we also need a generalization of the leftover hash lemma [13] that may be of independent interest.

Applications. One application of our method is centered around Damgård's public-key scheme [5] (from 1991) which he proved IND-CCA1 secure under the rather strong *knowledge of exponent* assumption.[1] This scheme can be viewed as a "double-base" variant of the original ElGamal encryption scheme [10] and consequently it is often referred to as *Damgård's ElGamal* in the literature. We first view the scheme as a hybrid encryption scheme (as advocated in [24,4]), applying our methodology of randomness extraction in the KEM's symmetric key derivation before the authenticated encryption (as DEM). The resulting scheme is a hybrid Damgård's ElGamal which is IND-CCA2 secure, under the standard DDH assumption. We furthermore propose a couple of variants of our basic hybrid scheme that offer certain efficiency tradeoffs. Compared to Cramer

[1] This assumption basically states that given two group elements (g_1, g_2) with unknown discrete logarithm $\omega = \log_{g_1}(g_2)$, the *only way* to efficiently compute (g_1^x, g_2^x) is to *know* the exponent x.

and Shoup's original scheme [2] and the improved scheme given by Kurosawa-Desmedt [17], our scheme crucially removes the dependence on the hard to construct target collision hash functions (UOWHF), using an easy-to-instantiate 4-wise independent hash function instead. Furthermore, the IND-CCA2 security of hybrid Damgård's ElGamal can be directly explained through our randomness extraction paradigm when applying it to the DDH-based universal$_1$ hash proof system. In contrast, due to the dependence on the target colission resistant hash function, the efficient schemes from [2,17] cannot be directly explained through Cramer and Shoup's hash proof system framework [3] and therefore all require separate proofs.

Another application of our method is given by a κ-entropic HPS from the QR assumption which is a variant of a HPS by Cramer and Shoup [3]. The resulting IND-CCA2 secure encryption scheme has very compact ciphertexts which only consist of one single element in $\mathbb{Z}_N^*$ plus the symmetric part. Like the scheme by Cramer and Shoup, the number of exponentiations in $\mathbb{Z}_N$ (for encryption and decryption) is linear in the security parameter. Hofheinz and Kiltz [15] give an IND-CCA2 secure encryption scheme based on the factoring assumption that is much more efficient than ours but has slightly larger ciphertexts.

Related Work. Cramer and Shoup [3] already proposed a generic transformation from universal$_1$ to universal$_2$ HPSs. Unfortunately their construction involves a significant overhead: the key of their transformed universal$_2$ HPS has linearly many keys of the original universal$_1$ HPS. We further remark that the notion of randomness extraction has had numerous applications in complexity and cryptography, and in particular in extracting random keys at the final step of key exchange protocols. Indeed, Cramer and Shoup [3] already proposed using a pairwise independent hash function to turn a universal$_1$ HPS into a universal$_2$ HPS. Our novel usage is within the context of hybrid encryption as a tool that shifts the integrity checking at decryption time solely to the DEM portion. In stark contrast to the generic transformations by Cramer and Shoup ours is practical.

Various previous proofs of variants of Damgård's original scheme have been suggested after Damgård himself proved it IND-CCA1 secure under the strong "knowledge of exponent" assumption (an assumption that has often been criticized in the literature; e.g., it is not efficiently falsifiable according to the classification of Naor [19]). More recent works are by Gjøsteen [12] who showed the scheme IND-CCA1 secure under some *interactive* version of the DDH assumption, where the adversary is given oracle access to some (restricted) DDH oracle. Also, Wu and Stinson [26], and at the same time Lipmaa [18] improve on the above two results. However, their security results are much weaker than ours: they only prove IND-CCA1 security of Damgård's ElGamal, still requiring security assumptions that are either interactive or of "knowledge of exponent" type. Desmedt and Hieu [8] recently showed a hybrid variant that is IND-CCA2 secure, yet under an even stronger assumption than Damgård's. Finally, and concurrently with our work, Desmedt et al. [7] recently showed a hybrid variant IND-CCA1 secure under the DDH assumption and a weaker KDF than ours.

2 Preliminaries

2.1 Notation

If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. If S is a set then $s \leftarrow_R S$ denotes the operation of picking an element s of S uniformly at random. We write $\mathsf{A}(x, y, \ldots)$ to indicate that A is an algorithm with inputs $x, y, \ldots$ and by $z \leftarrow_R \mathsf{A}(x, y, \ldots)$ we denote the operation of running A with inputs $(x, y, \ldots)$ and letting z be the output. We write $\lg x$ for logarithms over the reals with base 2. The *statistical distance* between two random variables X and Y having a common domain $\mathcal{X}$ is $\Delta[X, Y] = \frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr[X = x] - \Pr[Y = x]|$. We also define the conditional statistical distance as $\Delta_E[X, Y] = \frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr[X = x \mid E] - \Pr[Y = x \mid E]|$. The *min-entropy* of a random variable X is defined as $H_\infty(X) = -\lg(\max_{x\in\mathcal{X}} \Pr[X = x])$.

2.2 Public-Key Encryption

A *public key encryption* scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}(k)$ consists of three polynomial time algorithms (PTAs), of which the first two, Kg and Enc, are probabilistic and the last one, Dec, is deterministic. Public/secret keys for security parameter $k \in \mathbb{N}$ are generated using $(pk, sk) \leftarrow_R \mathsf{Kg}(1^k)$. Given such a key pair, a message $m \in \mathcal{M}(k)$ is encrypted by $C \leftarrow_R \mathsf{Enc}(pk, m)$; a ciphertext is decrypted by $m \leftarrow_R \mathsf{Dec}(sk, C)$, where possibly Dec outputs $\perp$ to denote an invalid ciphertext. For consistency, we require that for all $k \in \mathbb{N}$, all messages $m \in \mathcal{M}(k)$, it must hold that $\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1$ where the probability is taken over the above randomized algorithms and $(pk, sk) \leftarrow_R \mathsf{Kg}(1^k)$.

The security we require for PKE is $\mathsf{IND\text{-}CCA2}$ security [22,9]. We define the advantage of an adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ as

$$\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{PKE},\mathsf{A}}(k) \stackrel{\mathrm{def}}{=} \left| \Pr\left[b = b' \;:\; \begin{array}{l} (pk, sk) \leftarrow_R \mathsf{Kg}(1^k) \\ (m_0, m_1, St) \leftarrow_R \mathsf{A}_1^{\mathsf{Dec}(sk,\cdot)}(pk) \\ b \leftarrow_R \{0,1\};\ C^* \leftarrow_R \mathsf{Enc}(pk, m_b) \\ b' \leftarrow_R \mathsf{A}_2^{\mathsf{Dec}(sk,\cdot)}(C^*, St) \end{array} \right] - \frac{1}{2} \right| .$$

The adversary A_2 is restricted not to query $\mathsf{Dec}(sk, \cdot)$ with C^*. PKE scheme PKE is said to be indistinguishable against chosen-ciphertext attacks ($\mathsf{IND\text{-}CCA2}$ secure in short) if the advantage function $\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{PKE},\mathsf{A}}(k)$ is a negligible function in k for all adversaries $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ with probabilistic PTA A_1, A_2.

For integers k, t, Q we also define $\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{PKE},t,Q}(k) = \max_{\mathsf{A}} \mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{PKE},\mathsf{A}}(k)$, where the maximum is over all A that run in time at most t while making at most Q decryption queries.

We also mention the weaker security notion of *indistinguishability against lunch-time attacks* ($\mathsf{IND\text{-}CCA1}$ security), which is defined as $\mathsf{IND\text{-}CCA2}$ security with the restriction that the adversary is not allowed to make decryption queries after having seen the challenge ciphertext.

2.3 Hash Proof Systems

Smooth Projective Hashing. We recall the notion of hash proof systems as introduced by Cramer and Shoup [3]. Let $\mathcal{C}, \mathcal{K}$ be sets and $\mathcal{V} \subset \mathcal{C}$ a language. In the context of public-key encryption (and viewing a hash proof system as a key encapsulation mechanism (KEM) [4] with "special algebraic properties") one may think of $\mathcal{C}$ as the set of all *ciphertexts*, $\mathcal{V} \subset \mathcal{C}$ as the set of all *valid (consistent) ciphertexts*, and $\mathcal{K}$ as the set of all *symmetric keys*. Let $\Lambda_{sk} : \mathcal{C} \to \mathcal{K}$ be a hash function indexed with $sk \in \mathcal{SK}$, where $\mathcal{SK}$ is a set. A hash function Λ_{sk} is *projective* if there exists a projection $\mu : \mathcal{SK} \to \mathcal{PK}$ such that $\mu(sk) \in \mathcal{PK}$ defines the action of Λ_{sk} over the subset $\mathcal{V}$. That is, for every $C \in \mathcal{V}$, the value $K = \Lambda_{sk}(C)$ is uniquely determined by $\mu(sk)$ and C. In contrast, nothing is guaranteed for $C \in \mathcal{C} \setminus \mathcal{V}$, and it may not be possible to compute $\Lambda_{sk}(C)$ from $\mu(sk)$ and C. More precisely, following [14] we define universal$_1$ and universal$_2$ as follows.

universal$_1$. The projective hash function is ϵ_1-*almost universal*$_1$ if for all $C \in \mathcal{C} \setminus \mathcal{V}$,

$$\Delta\left[(pk, \Lambda_{sk}(C)\,,\ (pk, K)\right] \leq \epsilon_1 \tag{1}$$

where in the above $pk = \mu(sk)$ for $sk \leftarrow_R \mathcal{SK}$ and $K \leftarrow_R \mathcal{K}$.

universal$_2$. The projective hash function is ϵ_2-*almost universal*$_2$ if for all $C, C^* \in \mathcal{C} \setminus \mathcal{V}$ with $C \neq C^*$,

$$\Delta\left[(pk, \Lambda_{sk}(C^*), \Lambda_{sk}(C)\,,\ (pk, \Lambda_{sk}(C^*), K)\right] \leq \epsilon_2 \tag{2}$$

where in the above $pk = \mu(sk)$ for $sk \leftarrow_R \mathcal{SK}$ and $K \leftarrow_R \mathcal{K}$.

We introduce the following relaxation of the universal$_1$ property which only requires that for all $C \in \mathcal{C} \setminus \mathcal{V}$, given $pk = \mu(sk)$, $\Lambda_{sk}(C)$ has high min entropy.

κ-entropic. The projective hash function is ϵ_1-*almost* κ-*entropic* if for all $C \in \mathcal{C} \setminus \mathcal{V}$,

$$\Pr\left[\, H_\infty(\Lambda_{sk}(C) \mid pk) \geq \kappa \,\right] \geq 1 - \epsilon_1 \tag{3}$$

where in the above $pk = \mu(sk)$ for $sk \leftarrow_R \mathcal{SK}$.

From the above definitions, we get the following simple lemma.

Lemma 1. *Every ϵ_1-almost universal$_1$ projective hash function is ϵ_1-almost κ-entropic, for $\kappa = \lg(|\mathcal{K}|)$.*

Collision probability. To a projective hash function we also associate the collision probability, δ, defined as

$$\delta = \max_{C, C^* \in \mathcal{C} \setminus \mathcal{V}, C \neq C^*} \left(\Pr_{sk}\left[\, \Lambda_{sk}(C) = \Lambda_{sk}(C^*) \,\right]\right). \tag{4}$$

Hash Proof System. A hash proof system $\mathsf{HPS} = (\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ consists of three algorithms. The randomized algorithm $\mathsf{Param}(1^k)$ generates parametrized

instances of $params = (group, \mathcal{K}, \mathcal{C}, \mathcal{V}, \mathcal{PK}, \mathcal{SK}, \Lambda_{(\cdot)} : \mathcal{C} \to \mathcal{K}, \mu : \mathcal{SK} \to \mathcal{PK})$, where *group* may contain some additional structural parameters. The deterministic public evaluation algorithm Pub inputs the projection key $pk = \mu(sk), C \in \mathcal{V}$ and a witness r of the fact that $C \in \mathcal{V}$ and returns $K = \Lambda_{sk}(C)$. The deterministic private evaluation algorithm Priv inputs $sk \in \mathcal{SK}$ and returns $\Lambda_{sk}(C)$, without knowing a witness. We further assume that μ is efficiently computable and that there are efficient algorithms given for sampling $sk \in \mathcal{SK}$, sampling $C \in \mathcal{V}$ uniformly (or negligibly close to) together with a witness r, sampling $C \in \mathcal{C}$ uniformly, and for checking membership in $\mathcal{C}$.

We say that a hash proof system is universal$_1$ (resp., κ-entropic, universal$_2$) if for all possible outcomes of $\mathsf{Param}(1^k)$ the underlying projective hash function is $\epsilon_1(k)$-almost universal$_1$ (resp., $\epsilon_1(k)$-almost entropic, $\epsilon_2(k)$-almost universal$_2$) for negligible $\epsilon_1(k)$ (resp., $\epsilon_2(k)$). Furthermore, we say that a hash proof system is perfectly universal$_1$ (resp., κ-entropic, universal$_2$) if $\epsilon_1(k) = 0$ (resp., $\epsilon_2(k)$).

Subset Membership Problem. As computational problem we require that the *subset membership problem* is hard in HPS which means that for random $C_0 \in \mathcal{V}$ and random $C_1 \in \mathcal{C} \setminus \mathcal{V}$ the two elements C_0 and C_1 are computationally indistinguishable. This is captured by defining the advantage function $\mathsf{Adv}^{\mathrm{sm}}_{\mathsf{HPS},\mathsf{A}}(k)$ of an adversary A as

$$\mathsf{Adv}^{\mathrm{sm}}_{\mathsf{HPS},\mathsf{A}}(k) \stackrel{\mathrm{def}}{=} \left|\Pr\left[\,\mathsf{A}(\mathcal{C}, \mathcal{V}, C_1) = 1\,\right] - \Pr\left[\,\mathsf{A}(\mathcal{C}, \mathcal{V}, C_0) = 1\,\right]\right|$$

where $\mathcal{C}$ is taken from the output of $\mathsf{Param}(1^k)$, $C_1 \leftarrow_R \mathcal{C}$ and $C_0 \leftarrow_R \mathcal{C} \setminus \mathcal{V}$.

Hash Proof Systems with Trapdoor. Following [17], we also require that the subset membership problem can be efficiently solved with a master trapdoor. More formally, we assume that the hash proof system HPS additionally contains two algorithms Param' and Decide. The alternative parameter generator $\mathsf{Param}'(1^k)$ generates output indistinguishable from the one of $\mathsf{Param}(1^k)$ and additionally returns a trapdoor ω. The subset membership deciding algorithm $\mathsf{Decide}(params, \omega, x)$ returns 1 if $x \in \mathcal{V}$, and 0, otherwise. All known hash proof systems actually have such a trapdoor.

2.4 Symmetric Encryption

A symmetric encryption scheme $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ is specified by its encryption algorithm E (encrypting $m \in \mathcal{M}(k)$ with keys $S \in \mathcal{K}_{\mathsf{SE}}(k)$) and decryption algorithm D (returning $m \in \mathcal{M}(k)$ or $\perp$). Here we restrict ourselves to deterministic algorithms E and D.

The most common notion of security for symmetric encryption is that of (one-time) ciphertext indistinguishability ($\mathsf{IND\text{-}OT}$), which requires that all efficient adversaries fail to distinguish between the encryptions of two messages of their choice. Another common security requirement is *ciphertext authenticity*. (One-time) ciphertext integrity ($\mathsf{INT\text{-}OT}$) requires that no efficient adversary can produce a new valid ciphertext under some key when given one encryption of a message of his choice under the same key. A symmetric encryption scheme

which satisfies *both* requirements simultaneously is called secure in the sense of authenticated encryption (AE-OT secure). Note that AE-OT security is a stronger notion than chosen-ciphertext security. Formal definitions and constructions are provided in the full version [16]. There we also recall how to build a symmetric scheme with k-bit keys secure in the sense of AE-OT from a (computationally secure) one-time symmetric encryption scheme, a (computationally secure) MAC, and a (computationally secure) key-derivation function.

3 Randomness Extraction

In this section we review a few concepts related to probability distributions and extracting uniform bits from weak random sources. As a technical tool for our new paradigm, we will prove the following generalization of the leftover hash lemma [13]: if H is 4-wise independent, then $(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X}))$ is close to uniformly random, where $X, \tilde{X}$ can be dependent (but of course we have to require $X \neq \tilde{X}$).

Let $\mathcal{H}$ be a family of hash functions $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$. With $|\mathcal{H}|$ we denote the number of functions in this family and when sampling from $\mathcal{H}$ we assume a uniform distribution. Let $k > 1$ be an integer, the hash-family $\mathcal{H}$ is k-wise independent if for any sequence of distinct elements $x_1, \ldots, x_k \in \mathcal{X}$ the random variables $\mathsf{H}(x_1), \ldots, \mathsf{H}(x_k)$, where $\mathsf{H} \leftarrow_R \mathcal{H}$, are uniform random. We refer to [6] for a simple and efficient construction of a compressing k-wise independent hash function.

Recall that the leftover hash lemma states that for a 2-wise independent hash function H and a random variable X with min-entropy exceeding the bitlength of H's range, the random variable $(\mathsf{H}, \mathsf{H}(X))$ is close to uniformly random [13].

Lemma 2. *Let $X \in \mathcal{X}$ be a random variable where $H_\infty(X) \geq \kappa$. Let $\mathcal{H}$ be a family of pairwise independent hash functions with domain $\mathcal{X}$ and image $\{0,1\}^\ell$. Then for $\mathsf{H} \leftarrow_R \mathcal{H}$ and $U_\ell \leftarrow_R \{0,1\}^\ell$, $\Delta\left[(\mathsf{H}, \mathsf{H}(X)), (\mathsf{H}, U_\ell)\right] \leq 2^{(\ell-\kappa)/2}$.*

We will now prove a generalization of the leftover hash lemma that states that even when the hash function is evaluated in two distinct points, the two outputs jointly still look uniformly random. To make this work, we need a 4-wise independent hash function and, as before, sufficient min-entropy in the input distribution. We do note that, unsurprisingly, the loss of entropy compared to Lemma 2 is higher, as expressed in the bound on the statistical distance (or alternatively, in the bound on the min-entropy required in the input distribution).

Lemma 3. *Let $(X, \tilde{X}) \in \mathcal{X} \times \mathcal{X}$ be two random variables (having joint distribution) where $H_\infty(X) \geq \kappa, H_\infty(\tilde{X}) \geq \kappa$ and $\Pr[X = \tilde{X}] \leq \delta$. Let $\mathcal{H}$ be a family of 4-wise independent hash functions with domain $\mathcal{X}$ and image $\{0,1\}^\ell$. Then for $\mathsf{H} \leftarrow_R \mathcal{H}$ and $U_{2\ell} \leftarrow_R \{0,1\}^{2\ell}$,*

$$\Delta\left[(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})), (\mathsf{H}, U_{2\ell})\right] \leq \sqrt{1+\delta} \cdot 2^{\ell-\kappa/2} + \delta .$$

Proof. We will first prove the lemma for $\delta = 0$, and at the end show how the general case $\delta > 0$ can be reduced to it. Let $d = \lg |\mathcal{H}|$. For a random variable Y and Y' an independent copy of Y, we denote with $\mathsf{Col}[Y] = \Pr[Y = Y']$ the collision probability of Y. In particular,

$$\begin{aligned}
&\mathsf{Col}[(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X}))] \\
&= \Pr_{\mathsf{H},(X,\tilde{X}),\mathsf{H}',(X',\tilde{X}')}[(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})) = (\mathsf{H}', \mathsf{H}'(X'), \mathsf{H}'(\tilde{X}'))] \\
&= \Pr_{\mathsf{H},\mathsf{H}'}[\mathsf{H} = \mathsf{H}'] \cdot \Pr_{\mathsf{H},(X,\tilde{X}),\mathsf{H}',(X',\tilde{X}')}[(\mathsf{H}(X), \mathsf{H}(\tilde{X})) = (\mathsf{H}'(X'), \mathsf{H}'(\tilde{X}')) \mid \mathsf{H} = \mathsf{H}'] \\
&= \underbrace{\Pr_{\mathsf{H},\mathsf{H}'}[\mathsf{H} = \mathsf{H}']}_{=2^{-d}} \cdot \Pr_{\mathsf{H},(X,\tilde{X}),(X',\tilde{X}')}[(\mathsf{H}(X), \mathsf{H}(\tilde{X})) = (\mathsf{H}(X'), \mathsf{H}(\tilde{X}'))] \,. \qquad (5)
\end{aligned}$$

We define the event E, which holds if $X, \tilde{X}, X', \tilde{X}'$ are pairwise different.

$$\begin{aligned}
\Pr_{(X,\tilde{X}),(X',\tilde{X}')}[\neg\mathrm{E}] &= \Pr_{(X,\tilde{X}),(X',\tilde{X}')}[X = X' \vee X = \tilde{X}' \vee \tilde{X} = X' \vee \tilde{X} = \tilde{X}'] \\
&\leq 4 \cdot 2^{-\kappa} = 2^{-\kappa+2}
\end{aligned}$$

Where in the first step we used that $\delta = 0$, and thus $X \neq \tilde{X}, X' \neq \tilde{X}'$. In the second step we use the union bound and also our assumption that the min entropy of X and $\tilde{X}$ is at least κ (and thus, e.g., $\Pr[X = X'] \leq 2^{-\kappa}$). With this we can write (5) as

$$\begin{aligned}
\mathsf{Col}[\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})] &\leq 2^{-d} \cdot (\Pr[(\mathsf{H}(X), \mathsf{H}(\tilde{X})) = (\mathsf{H}(X'), \mathsf{H}(\tilde{X}')) \mid \mathrm{E}] + \Pr[\neg\mathrm{E}]) \\
&\leq 2^{-d}(2^{-2\ell} + 2^{-\kappa+2})
\end{aligned}$$

where in the second step we used that H is 4-wise independent. Let Y be a random variable with support $\mathcal{Y}$ and U be uniform over $\mathcal{Y}$, then

$$\|Y - U\|_2^2 = \mathsf{Col}[Y] - |\mathcal{Y}|^{-1} \,.$$

In particular,

$$\begin{aligned}
\|(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})) - (\mathsf{H}, U_{2\ell})\|_2^2 &= \mathsf{Col}[\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})] - 2^{-d-2\ell} \\
&\leq 2^{-d}(2^{-2\ell} + 2^{-\kappa+2}) - 2^{-d-2\ell} = 2^{-d-\kappa+2} \,.
\end{aligned}$$

Using that $\|Y\|_1 \leq \sqrt{|\mathcal{Y}|}\|Y\|_2$ for any random variable Y with support $\mathcal{Y}$, we obtain

$$\begin{aligned}
\Delta\left[(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})), (\mathsf{H}, U_{2\ell})\right] &= \frac{1}{2}\|(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})) - (\mathsf{H}, U_{2\ell})\|_1 \\
&\leq \frac{1}{2}\sqrt{2^{d+2\ell}} \cdot \|(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})) - (\mathsf{H}, U_{2\ell})\|_2 \\
&\leq \frac{1}{2}\sqrt{2^{d+2\ell}} \cdot \sqrt{2^{-d-\kappa+2}} = 2^{\ell-\kappa/2} \,.
\end{aligned}$$

This concludes the proof of (3) for $\delta = 0$. Now consider $X, \tilde{X}$ as in the statement of the lemma where $\Pr[X = \tilde{X}] \leq \delta$ for some $\delta > 0$. Let π denote any permutation over $\mathcal{X}$ without a fixpoint, i.e., $\pi(x) \neq x$ for all $x \in \mathcal{X}$. Let $(Y, \tilde{Y})$ be sampled as follows: first sample $(X, \tilde{X})$, if $X \neq \tilde{X}$ let $(Y, \tilde{Y}) = (X, \tilde{X})$, otherwise sample $Y \leftarrow_R \mathcal{X}$ uniformly at random and set $\tilde{Y} := \pi(Y)$. By definition $\Pr[Y = \tilde{Y}] = 0$, and as $(Y, \tilde{Y})$ has the same distribution as $(X, \tilde{X})$ except with probability δ, $\Delta\left[(X, \tilde{X}), (Y, \tilde{Y})\right] \leq \delta$. Moreover, using that $\max_{x \in \mathcal{X}} \Pr[X = x] \leq 2^{-\kappa}$

$$\max_{x \in \mathcal{X}} \Pr[Y = x] \leq 2^{-\kappa} + \delta/|\mathcal{X}| \leq (1 + \delta)2^{-\kappa} .$$

Thus $H_\infty(Y) \geq \kappa - \lg(1 + \delta)$, and similarly $H_\infty(\tilde{Y}) \geq \kappa - \lg(1 + \delta)$. We can now apply the lemma for the special case $\delta = 0$ (which we proved) and get

$$\Delta\left[(\mathsf{H}, \mathsf{H}(Y), \mathsf{H}(\tilde{Y})), (\mathsf{H}, U_{2\ell})\right] \leq 2^{\ell - (\kappa - \lg(1+\delta))/2} = \sqrt{1 + \delta} \cdot 2^{\ell - \kappa/2} .$$

The lemma now follows as

$$\begin{aligned}
&\Delta\left[(\mathsf{H}, \mathsf{H}(X), \mathsf{H}(\tilde{X})), (\mathsf{H}, U_{2\ell})\right] \\
&\leq \Delta\left[(\mathsf{H}, \mathsf{H}(Y), \mathsf{H}(\tilde{Y})), (\mathsf{H}, U_{2\ell})\right] + \Delta\left[(X, \tilde{X}), (Y, \tilde{Y})\right] \\
&\leq \sqrt{1 + \delta} \cdot 2^{\ell - \kappa/2} + \delta .
\end{aligned}$$

4 Hybrid Encryption from Randomness Extraction

In this section we revisit the general construction of hybrid encryption from universal$_2$ hash proof systems. As our main technical result we show an efficient transformation from a κ-entropic to a universal$_2$ HPS, so in particular also from a universal$_1$ to a universal$_2$ HPS. Combining the latter universal$_2$ HPS with an $\mathsf{AE\text{-}OT}$ secure symmetric cipher gives an $\mathsf{IND\text{-}CCA2}$ secure hybrid encryption scheme. This result can be readily applied to all known hash proof systems with a hard subset membership problem that are universal$_1$ (e.g., from Paillier's DCR, the DDH/n-Linear [14,23] assumptions) or κ-entropic (e.g., from the QR [3] assumption) to obtain a number of new $\mathsf{IND\text{-}CCA2}$ secure hybrid encryption schemes. More concretely, in Section 5 we will discuss the consequences for DDH-based schemes and in Section 6 for QR-based schemes.

4.1 Hybrid Encryption from HPSs

Recall the notion of a hash proof system from Section 2.3. Kurosawa and Desmedt [17] proposed the following hybrid encryption scheme which improved the schemes from Cramer and Shoup [3].

Let $\mathsf{HPS} = (\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ be a hash proof system and let $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be an $\mathsf{AE\text{-}OT}$ secure symmetric encryption scheme whose key-space $\mathcal{K}_{\mathsf{SE}}$ matches

the key-space $\mathcal{K}$ of the HPS.[2] The system parameters of the scheme consist of $params \leftarrow_R \mathsf{Param}(1^k)$.

$\mathsf{Kg}(k)$. Choose random $sk \leftarrow_R \mathcal{SK}$ and define $pk = \mu(sk) \in \mathcal{PK}$. Return (pk, sk).

$\mathsf{Enc}(pk, m)$. Pick $C \leftarrow_R \mathcal{V}$ together with its witness r that $C \in \mathcal{V}$. The session key $K = \Lambda_{sk}(C) \in \mathcal{K}$ is computed as $K \leftarrow \mathsf{Pub}(pk, C, r)$. The symmetric ciphertext is $\psi \leftarrow \mathsf{E}_K(m)$. Return the ciphertext (C, ψ).

$\mathsf{Dec}(sk, C)$. Reconstruct the key $K = \Lambda_{sk}(C)$ as $K \leftarrow \mathsf{Priv}(sk, C)$ and return $\{m, \perp\} \leftarrow \mathsf{D}_K(\psi)$.

Note that the trapdoor property of the HPS is not used in the actual scheme: it is only needed in the proof. However, as an alternative the trapdoor can be added to the secret key.[3] This allows explicit rejection of invalid ciphertexts during decryption. The security of this explicit-rejection variant is identical to that of the scheme above.

The following was proved in [17,11,14].

Theorem 4. *Assume* HPS *is* (ϵ_2) *universal*$_2$ *with hard subset membership problem (with trapdoor), and* SE *is* AE-OT *secure. Then the encryption scheme is secure in the sense of* IND-CCA2*. In particular,*

$$\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{PKE},t,Q}(k) \leq \mathsf{Adv}^{\mathrm{sm}}_{\mathsf{HPS},t}(k) + 2Q \cdot \mathsf{Adv}^{\text{int-ot}}_{\mathsf{SE},t}(k) + \mathsf{Adv}^{\text{ind-ot}}_{\mathsf{SE},t}(k) + Q \cdot \epsilon_2 \ .$$

We remark that even though in general the KEM part of the above scheme cannot be proved IND-CCA2 secure [1], it can be proved "IND-CCCA" secure. The latter notion was defined in [14] and proved sufficient to yield IND-CCA2 secure encryption when combined with an AE-OT secure cipher. We also remark that the security bound in the above theorem implicitly requires that the image of $\Lambda_{sk}(\cdot)$ restricted to $\mathcal{V}$ is sufficiently large (say, contains at least 2^k elements). This is since otherwise the key-space of the symmetric scheme is too small and the two advantages functions $\mathsf{Adv}^{\text{int-ot}}_{\mathsf{SE},t}(k)$ and $\mathsf{Adv}^{\text{ind-ot}}_{\mathsf{SE},t}(k)$ cannot be negligible.

There is also an analogue "lite version" for universal$_1$ HPS, giving IND-CCA1 only (and using a slightly weaker asymmetric primitive). It can be stated as follows.

Theorem 5. *Assume* HPS *is universal*$_1$ *with hard subset membership problem and* SE *is* WAE-OT *secure. Then the encryption scheme is secure in the sense of* IND-CCA1*.*

We note that if the HPS is only κ-entropic then we can use the standard Leftover Hash Lemma (Lemma 2) to obtain a universal$_1$ HPS.

4.2 A Generic Transformation from κ-Entropic to Universal$_2$ HPSs

We propose the following transformation. Given a projective hash function $\Lambda_{sk} : \mathcal{C} \to \mathcal{K}$ with projection $\mu : \mathcal{SK} \to \mathcal{PK}$ and a family of hash functions $\mathcal{H}$ with $\mathsf{H} : \mathcal{K} \to \{0,1\}^\ell$. Then we define the hashed variant of it as:

[2] The requirement that $\mathcal{K}_{\mathsf{SE}} = \mathcal{K}$ is not a real restriction since one can always apply a key-derivation function $\mathsf{KDF} : \mathcal{K} \to \mathcal{K}_{\mathsf{SE}}$.

[3] Strictly speaking the algorithm to sample elements in $\mathcal{V}$ (with witness) should then be regarded as part of the public key instead of simply a system parameter.

$$\Lambda_{sk}^{\mathcal{H}} : \mathcal{C} \to \{0,1\}^{\ell}, \quad \Lambda_{sk}^{\mathcal{H}}(C) := \mathsf{H}(\Lambda_{sk}(C)) \ .$$

We also define $\mathcal{PK}^{\mathcal{H}} = \mathcal{PK} \times \mathcal{H}$ and $\mathcal{SK}^{\mathcal{H}} = \mathcal{SK} \times \mathcal{H}$, such that the hashed projection is given by $\mu^{\mathcal{H}} : \mathcal{SK}^{\mathcal{H}} \to \mathcal{PK}^{\mathcal{H}}$, $\mu^{\mathcal{H}}(sk, \mathsf{H}) = (pk, \mathsf{H})$. This also induces a transformation from a hash proof system HPS into $\mathsf{HPS}^{\mathcal{H}}$, where the above transformation is applied to the projective hash function. Note that $\mathcal{C}$ and $\mathcal{V}$ are the same for HPS and $\mathsf{HPS}^{\mathcal{H}}$ (so that in particular the trapdoor property for the language $\mathcal{V}$ is inherited).

We are now ready to state our main theorem. To simplify the bounds, we will henceforth assume that $\delta \leq \frac{1}{2}$ and $\ell \geq 6$.

Theorem 6. *Assume* HPS *is* ϵ_1*-almost* κ*-entropic with collision probability* $\delta \leq 1/2$ *and* $\mathcal{H}$ *is a family of 4-wise independent hash functions with* $\mathsf{H} : \mathcal{K} \to \{0,1\}^{\ell}$ *and* $\ell \geq 6$*. Then* $\mathsf{HPS}^{\mathcal{H}}$ *is* ϵ_2*-almost universal*$_2$ *for*

$$\epsilon_2 = 2^{\ell - \frac{\kappa - 1}{2}} + 3\epsilon_1 + \delta \ .$$

Proof. Let us consider, for all $C, C^* \in \mathcal{C} \setminus \mathcal{V}$ with $C \neq C^*$, the statistical distance relevant for universal$_2$ for HPS and let Y be the following random variable

$$Y := (pk, \mathsf{H}, U_{2\ell}) \ ,$$

where $pk = \mu(sk)$ for $sk \leftarrow_R \mathcal{SK}$, $\mathsf{H} \leftarrow_R \mathcal{H}$ and $U_{2\ell} \leftarrow_R \{0,1\}^{2\ell}$. Then we can use the triangle inequality to get

$$\begin{aligned} &\Delta\left[(pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), \mathsf{H}(\Lambda_{sk}(C)) \ , \ (pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), U_{\ell})\right] \\ &\leq \Delta\left[(pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), \mathsf{H}(\Lambda_{sk}(C))) \ , \ Y\right] + \Delta\left[Y \ , \ (pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), U_{\ell})\right] \quad (6) \end{aligned}$$

where as before $pk = \mu(sk)$ for $sk \leftarrow_R \mathcal{SK}$, $\mathsf{H} \leftarrow_R \mathcal{H}$ and $U_{\ell} \leftarrow_R \{0,1\}^{\ell}$. In the latter probability space, let E_{C^*} be the event that $H_{\infty}(\Lambda_{sk}(C^*) \mid pk) \geq \kappa$. We can upper bound the second term of (6), using again the triangle inequality in the first step, as

$$\begin{aligned} \Delta\left[Y \ , \ (pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), U_{\ell})\right] &\leq \Delta_{E_{C^*}}\left[Y \ , \ (pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), U_{\ell})\right] + \Pr_{sk}[\neg E_{C^*}] \\ &\leq 2^{\frac{\ell - \kappa}{2}} + \epsilon_1 \ . \quad (7) \end{aligned}$$

In the last step we used the (standard) leftover hash-lemma (Lemma 2). Let E_C be the event that $H_{\infty}(\Lambda_{sk}(C)) \mid pk) \geq \kappa$. Similarly, we now bound the first term of (6) as

$$\begin{aligned} &\Delta\left[(pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), \mathsf{H}(\Lambda_{sk}(C))) \ , \ Y\right] \\ &\leq \Delta_{E_C \wedge E_{C^*}}\left[(pk, \mathsf{H}, \mathsf{H}(\Lambda_{sk}(C^*)), \mathsf{H}(\Lambda_{sk}(C))) \ , \ Y\right] + \Pr_{sk}[\neg E_C \vee \neg E_{C^*}] \\ &\leq \sqrt{1+\delta} \cdot 2^{\ell - \frac{\kappa}{2}} + \delta + 2\epsilon_1 \ , \end{aligned}$$

where in the last step we used Lemma 3. Combining this with (7) and using $\delta \leq 1/2$ and $\ell \geq 6$ we obtain the required bound on ϵ_2.

4.3 Hybrid Encryption from κ-Entropic HPSs

Putting the pieces from the last two sections together we get a new IND-CCA2 secure hybrid encryption scheme from any κ-entropic hash proof system. Let $\mathsf{HPS} = (\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ be a hash proof system, let $\mathcal{H}$ be a family of hash functions with $\mathsf{H} : \mathcal{K} \to \{0,1\}^\ell$ and let $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be an AE-OT secure symmetric encryption scheme with key-space $\mathcal{K}_{\mathsf{SE}} = \{0,1\}^\ell$. The system parameters of the scheme consist of $params \leftarrow_R \mathsf{Param}(1^k)$.

$\mathsf{Kg}(k)$**.** Choose random $sk \leftarrow_R \mathcal{SK}$ and define $pk = \mu(sk) \in \mathcal{PK}$. Pick a random hash function $\mathsf{H} \leftarrow_R \mathcal{H}$. The public-key is (H, pk), the secret-key is (H, sk).

$\mathsf{Enc}(pk, m)$**.** Pick $C \leftarrow_R \mathcal{V}$ together with its witness r that $C \in \mathcal{V}$. The session key $K = \mathsf{H}(\Lambda_{sk}(C)) \in \{0,1\}^\ell$ is computed as $K \leftarrow \mathsf{H}(\mathsf{Pub}(pk, C, r))$. The symmetric ciphertext is $\psi \leftarrow \mathsf{E}_K(m)$. Return the ciphertext (C, ψ).

$\mathsf{Dec}(sk, C)$**.** Reconstruct the key $K = \mathsf{H}(\Lambda_{sk}(C))$ as $K \leftarrow \mathsf{H}(\mathsf{Priv}(sk, C))$ and return $\{m, \perp\} \leftarrow \mathsf{D}_K(\psi)$.

Combining Theorems 4 and 6 gives us the following corollary.

Corollary 7. *Assume* HPS *is (ϵ_1-almost) κ-entropic with hard subset membership problem and with collision probability $\delta(k)$, that $\mathcal{H}$ is a family of 4-wise independent hash functions with $\mathsf{H} : \mathcal{K} \to \{0,1\}^{\ell(k)}$, and that* SE *is* AE-OT *secure. If $2^{\ell(k)-\kappa(k)/2}$ and $\delta(k)$ are negligible, then the encryption scheme above is secure in the sense of* IND-CCA2*. In particular,*

$$\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{PKE},t,Q}(k) \le \mathsf{Adv}^{\mathrm{sm}}_{\mathsf{HPS},t}(k) + 2Q \cdot \mathsf{Adv}^{\text{int-ot}}_{\mathsf{SE},t}(k) + \mathsf{Adv}^{\text{ind-ot}}_{\mathsf{SE},t}(k) + Q \cdot (2^{\ell - \frac{\kappa-1}{2}} + 3\epsilon_1 + \delta).$$

5 Instantiations from the DDH Assumption

In this section we discuss two practical instantiations of our randomness extraction framework whose security is based on the DDH assumption.

5.1 The Decisional Diffie-Hellman (DDH) Assumption

A group scheme $\mathcal{GS}$ [4] specifies a sequence $(\mathcal{GR}_k)_{k\in\mathbb{N}}$ of group descriptions. For every value of a security parameter $k \in \mathbb{N}$, the pair $\mathcal{GR}_k = (\mathbb{G}_k, p_k)$ specifies a cyclic (multiplicative) group $\mathbb{G}_k$ of prime order p_k. Henceforth, for notational convenience, we tend to drop the index k. We assume the existence of an efficient sampling algorithm $x \leftarrow_R \mathbb{G}$ and an efficient membership algorithm. We define the ddh-advantage of an adversary B as

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{GS},\mathsf{B}}(k) \stackrel{\mathrm{def}}{=} \left|\Pr[\mathsf{B}(g_1, g_2, g_1^r, g_2^r) = 1] - \Pr[\mathsf{B}(g_1, g_2, g_1^r, g_2^{\tilde{r}}) = 1]\right|,$$

where $g_1,\ g_2 \leftarrow_R \mathbb{G}$, $r \leftarrow_R \mathbb{Z}_p$, $\tilde{r} \leftarrow_R \mathbb{Z}_p \setminus \{r\}$. We say that the DDH problem is hard in $\mathcal{GS}$ if the advantage function $\mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{GS},\mathsf{B}}(k)$ is a negligible function in k for all probabilistic PTA B.

5.2 Variant 1: The Scheme HE_1

THE UNIVERSAL$_1$ HASH PROOF SYSTEM. We recall a universal$_1$ HPS by Cramer and Shoup [3], whose hard subset membership problem is based on the DDH assumption. Let $\mathcal{GS}$ be a group scheme where $\mathcal{GR}_k$ specifies $(\mathbb{G}, p)$ and let g_1, g_2 be two independent generators of $\mathbb{G}$. Define $\mathcal{C} = \mathbb{G}^2$ and $\mathcal{V} = \{(g_1^r, g_2^r) \subset \mathbb{G}^2 : r \in \mathbb{Z}_p\}$. The value $r \in \mathbb{Z}_p$ is a witness of $C \in \mathcal{V}$. The trapdoor generator Param picks a uniform trapdoor $\omega \in \mathbb{Z}_p$ and computes $g_2 = g_1^\omega$. Note that using trapdoor ω, algorithm Decide can efficiently perform subset membership tests for $C = (c_1, c_2) \in \mathcal{C}$ by checking whether $c_1^\omega = c_2$.

Let $\mathcal{SK} = \mathbb{Z}_p^2$, $\mathcal{PK} = \mathbb{G}$, and $\mathcal{K} = \mathbb{G}$. For $sk = (x_1, x_2) \in \mathbb{Z}_p^2$, define $\mu(sk) = X = g_1^{x_1} g_2^{x_2}$. This defines the output of $\mathsf{Param}(1^k)$. For $C = (c_1, c_2) \in \mathcal{C}$ define

$$\Lambda_{sk}(C) := c_1^{x_1} c_2^{x_2} \ . \tag{8}$$

This defines $\mathsf{Priv}(sk, C)$. Given $pk = \mu(sk) = X$, $C \in \mathcal{V}$ and a witness $r \in \mathbb{Z}_p$ such that $C = (g_1^r, g_2^r)$ public evaluation $\mathsf{Pub}(pk, C, r)$ computes $K = \Lambda_{sk}(C)$ as

$$K = X^r \ .$$

Correctness follows by (8) and the definition of μ. This completes the description of HPS. Clearly, under the DDH assumption, the subset membership problem is hard in HPS. Moreover, this HPS is known to be (perfect) universal$_1$ [3].

Lemma 8. *The above HPS is perfect universal$_1$ (so $\epsilon_1 = 0$) with collision probability $\delta = 1/p$.*

Proof. To show that the HPS is universal$_1$, it suffices to show that given the public key X and any pair $(C, K) \in (\mathcal{C} \setminus \mathcal{V}) \times \mathcal{K}$, there exists exactly one secret key sk such that $\mu(sk) = X$ and $\Lambda_{sk}(C) = K$. Let $\omega \in \mathbb{Z}_p^*$ be such that $g_2 = g_1^\omega$, write $C = (g_1^r, g_2^s)$ for $r \neq s$ and consider a possible secret key $sk = (x_1, x_2) \in \mathbb{Z}_p^2$. Then we simultaneously need that $\mu(sk) = g_1^{x_1 + \omega x_2} = X = g^x$ (for some $x \in \mathbb{Z}_p$) and $\Lambda_{sk}(C) = g_1^{r x_1 + s\omega x_2} = K = g_1^y$ (for some $y \in \mathbb{Z}_p$). Then, using linear algebra, x_1 and x_2 follow uniquely from r, s, x, y and ω provided that the relevant determinant $(s - r)\omega \neq 0$. This is guaranteed here since $r \neq s$ and $\omega \neq 0$.

To verify the bound on the collision probability δ it suffices —due to symmetry— to determine for any distinct pair $(C, C^*) \in (\mathcal{C} \setminus \mathcal{V})^2$ the probability $\Pr_{sk}[\Lambda_{sk}(C) = \Lambda_{sk}(C^*)]$. In other words, for $(r, s) \neq (r', s')$ (with $r \neq s$ and $r' \neq s'$, but that is irrelevant here) we have that

$$\begin{aligned}
\delta &= \Pr_{x_1, x_2 \leftarrow_R \mathbb{Z}_p} [g_1^{r x_1 + x_2 \omega s} = g_1^{r' x_1 + x_2 \omega s'}] \\
&= \Pr_{x_1, x_2 \leftarrow_R \mathbb{Z}_p} [r x_1 + x_2 \omega s = r' x_1 + x_2 \omega s'] \\
&= 1/p \ .
\end{aligned}$$

(For the last step, use that if $r \neq r'$ for any x_2 only one x_1 will "work"; if $r = r'$ then necessarily $s \neq s'$ and for any x_1 there is a unique x_2 to satisfy the equation.)

THE HYBRID ENCRYPTION SCHEME HE_1. We apply the transformation from Section 4.3 to the above HPS and obtain an hybrid encryption scheme which is depicted in Figure 1.

Theorem 9. *Let $\mathcal{GS} = (\mathbb{G}, p)$ be a group scheme where the DDH problem is hard, let $\mathcal{H}$ be a family of 4-wise independent hash functions from $\mathbb{G}$ to $\{0,1\}^{\ell(k)}$ with $\lg p \geq 4\ell(k)$, and let SE be a symmetric encryption scheme with key-space $\mathcal{K}_{\mathsf{SE}} = \{0,1\}^{\ell(k)}$. that is secure in the sense of $\mathsf{AE\text{-}OT}$. Then HE_1 is secure in the sense of $\mathsf{IND\text{-}CCA2}$. In particular,*

$$\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{HE}_1,t,Q}(k) \leq \mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{GS},t}(k) + 2Q \cdot \mathsf{Adv}^{\text{int-ot}}_{\mathsf{SE},t}(k) + \mathsf{Adv}^{\text{ind-ot}}_{\mathsf{SE},t}(k) + Q \cdot 2^{-\ell(k)+1} \ .$$

Proof. By Lemma 8 the HPS is (perfectly) universal$_1$ and therefore (by Lemma 1) it is also (perfectly) κ-entropic with $\kappa = \lg(|\mathcal{K}|) = \lg p \geq 4\ell(k)$. It leaves to bound the loss due to the κ-entropic to universal$_2$ HPS transformation from Corollary 7:

$$(1+\delta)2^{\ell - \frac{\kappa}{2}} + 2^{\frac{\ell-\kappa}{2}} + 3\epsilon_1 + \delta \leq 2^{-\ell+1}$$

where we used that $|\mathcal{K}| = |\mathbb{G}| = p \geq 2^{4\ell}$ and (by Lemma 8) $\epsilon_1 = 0$ and $\delta = 1/p$.

We remark that in terms of concrete security, Theorem 9 requires the image $\{0,1\}^{\ell(k)}$ of H to be sufficiently small, i.e., $\ell(k) \leq \frac{1}{4}\lg p$. For a symmetric cipher with $\ell(k) = k = 80$ bits keys we are forced to use groups of order $\lg p = 4k = 320$ bits. For some specific groups such as elliptic curves this can be a drawback since there one typically works with groups of order $\lg p = 2k = 160$ bits.

$\mathsf{Kg}(1^k)$	$\mathsf{Enc}(pk, m)$	$\mathsf{Dec}(sk, C)$
$x_1, x_2 \leftarrow_R \mathbb{Z}_p$; $X \leftarrow g_1^{x_1} g_2^{x_2}$	$r \leftarrow_R \mathbb{Z}_p^*$	Parse C as (c_1, c_2, ψ)
Pick $\mathsf{H} \leftarrow_R \mathcal{H}$	$c_1 \leftarrow g_1^r$; $c_2 \leftarrow g_2^r$	$K \leftarrow \mathsf{H}(c_1^{x_1} c_2^{x_2})$
$pk \leftarrow (X, \mathsf{H})$; $sk \leftarrow (x_1, x_2)$	$K \leftarrow \mathsf{H}(X^r) \in \{0,1\}^\ell$	Return $\{m, \perp\} \leftarrow \mathsf{D}_K(\psi)$
Return (sk, pk)	$\psi \leftarrow \mathsf{E}_K(m)$	
	Return $C = (c_1, c_2, \psi)$	

Fig. 1. Hybrid encryption scheme $\mathsf{HE}_1 = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ obtained by applying our randomness extraction framework to the HPS from Section 5.2

RELATION TO DAMGÅRD'S ELGAMAL. In HE_1, invalid ciphertexts of the form $c_1^\omega \neq c_2$ are rejected implicitly by authenticity properties of the symmetric cipher. Similar to [4], a variant of this scheme, $\mathsf{HE}_1^{\mathsf{er}} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$, in which such invalid ciphertexts get explicitly rejected is given in Figure 2. The scheme is slightly simplified compared to a direct explicit version that adds the trapdoor to the secret key; the simplification can be justified using the techniques of Lemma 8.

We remark that, interestingly, Damgård's encryption scheme [5] (also known as Damgård's ElGamal) is a special case of $\mathsf{HE}_1^{\mathsf{er}}$ from Fig. 2 where the hash function H is the identity function (or an easy-to-invert, canonical embedding of

$\mathsf{Kg}(1^k)$	$\mathsf{Enc}(pk, m)$	$\mathsf{Dec}(sk, C)$
$\omega, x \leftarrow_R \mathbb{Z}_p$	$r \leftarrow_R \mathbb{Z}_p^*$	Parse C as (c_1, c_2, ψ)
$g_2 \leftarrow g_1^{\omega}$; $X \leftarrow g_1^x$	$c_1 \leftarrow g_1^r$; $c_2 \leftarrow g_2^r$	if $c_1^{\omega} \neq c_2$ return $\perp$
Pick $\mathsf{H} \leftarrow_R \mathcal{H}$	$K \leftarrow \mathsf{H}(X^r)$	$K \leftarrow \mathsf{H}(c_1^x)$
$pk \leftarrow (g_2, X, \mathsf{H})$; $sk \leftarrow (x, \omega)$	$\psi \leftarrow \mathsf{E}_K(m)$	Return $\{m, \perp\} \leftarrow \mathsf{D}_K(\psi)$
Return (sk, pk)	Return $C = (c_1, c_2, \psi)$	

Fig. 2. Hybrid encryption scheme $\mathsf{HE}_1^{\mathsf{er}} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. A variant of HE_1 with explicit rejection

the group into, say, the set of bitstrings) and SE is "any easy to invert group operation" [5], for example the one-time pad with $\mathsf{E}_K(m) = K \oplus m$. In his paper, Damgård proved IND-CCA1 security of his scheme under the DDH assumption and the *knowledge of exponent* assumption in $\mathcal{GS}$.[4] Our schemes $\mathsf{HE}_1^{\mathsf{er}}$ and HE_1 can therefore be viewed as hybrid versions of Damgård's ElGamal scheme, that can be proved IND-CCA2 secure under the DDH assumption.

5.3 Variant 2: The Scheme HE_2

The universal$_1$ hash proof system. We now give an alternative (and new) universal$_1$ hash proof system from the DDH assumption. Keep $\mathcal{C}$ and $\mathcal{V}$ as in Section 5.2. Define $\mathcal{SK} = \mathbb{Z}_p^4$, $\mathcal{PK} = \mathbb{G}^2$, and $\mathcal{K} = \mathbb{G}^2$. For $sk = (x_1, x_2, \hat{x}_1, \hat{x}_2) \in \mathbb{Z}^4$, define $\mu(sk) = (X, \hat{X}) = (g_1^{x_1} g_2^{x_2}, g_1^{\hat{x}_1} g_2^{\hat{x}_2})$. For $C = (c_1, c_2) \in \mathcal{C}$ define

$$\Lambda_{sk}(C) := (c_1^{x_1} c_2^{x_2}, c_1^{\hat{x}_1} c_2^{\hat{x}_2}) \; .$$

This also defines $\mathsf{Priv}(sk, C)$. Given $pk = \mu(sk)$, $C \in \mathcal{V}$ and a witness $r \in \mathbb{Z}_p$ such that $C = (c_1, c_2) = (g_1^r, g_2^r)$, public evaluation $\mathsf{Pub}(pk, C, r)$ computes $K = \Lambda_{sk}(C)$ as

$$K = (X^r, \hat{X}^r) \; .$$

Similar to Lemma 8 we can prove the following.

Lemma 10. *The above HPS is perfect universal$_1$ with collision probability $\delta = 1/p^2$.*

The scheme HE_2. For our second hybrid encryption scheme HE_2 we make the same assumption as for HE_1, with the difference that $\mathcal{H}$ is now a family $\mathsf{H}_k : \mathbb{G}^2 \to \{0,1\}^{\ell(k)}$ of 4-wise independent hash functions with $\lg p \geq 2\ell(k)$. The resulting hybrid encryption scheme obtained by applying Corollary 7 (in conjuction with Lemma 10) is depicted in Figure 3.

[4] To be more precise, Damgård only formally proved one-way (OW-CCA1) security of his scheme, provided that the original ElGamal scheme is OW-CPA secure. But he also remarks that his proof can be reformulated to prove IND-CCA1 security, provided that ElGamal itself is IND-CPA secure. IND-CPA security of ElGamal under the DDH assumption was only formally proved later [25].

Theorem 11. *Let $\mathcal{GS} = (\mathbb{G}, p)$ be a group scheme where the DDH problem is hard, let $\mathcal{H}$ be a family of 4-wise independent hash functions from $\mathbb{G}^2$ to $\{0,1\}^{\ell(k)}$ with $\lg p \geq 2\ell(k)$, and let SE be a symmetric encryption scheme with key-space $\mathcal{K}_{\mathsf{SE}} = \{0,1\}^{\ell(k)}$ that is secure in the sense of* AE-OT*. Then HE_2 is secure in the sense of* IND-CCA2*. In particular,*

$$\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{HE}_2,t,Q}(k) \leq \mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{GS},t}(k) + 2Q \cdot \mathsf{Adv}^{\text{int-ot}}_{\mathsf{SE},t}(k) + \mathsf{Adv}^{\text{ind-ot}}_{\mathsf{SE},t}(k) + Q \cdot 2^{-\ell(k)+1} \ .$$

Note that HE_2 now only has the restriction $\lg p \geq 2\ell(k)$ which fits nicely with the typical choice of $\ell(k) = k$ and $\lg p = 2k$. So one is free to use any cryptographic group, in particular also elliptic curve groups.

Similar to $\mathsf{HE}_1^{\mathsf{er}}$, the variant $\mathsf{HE}_2^{\mathsf{er}}$ with explicit rejection can again be proven equivalent. In the explicit rejection variant, $\mathsf{HE}_2^{\mathsf{er}}$, the public-key contains the group elements $g_2 = g_1^{\omega}$, $X = g_1^{x}$, and $\hat{X} = g_1^{\hat{x}}$), and decryption first checks if $c_1^{\omega} = c_2$ and then computes $K = \mathsf{H}(c_1^{x}, c_1^{\hat{x}})$.

$\mathsf{Kg}(1^k)$	$\mathsf{Enc}(pk, m)$	$\mathsf{Dec}(sk, C)$
$x_1, x_2, \hat{x}_1, \hat{x}_2 \leftarrow_R \mathbb{Z}_p$	$r \leftarrow_R \mathbb{Z}_p^*$	Parse C as (c_1, c_2, ψ)
$X \leftarrow g_1^{x_1} g_2^{x_2}$; $\hat{X} \leftarrow g_1^{\hat{x}_1} g_2^{\hat{x}_2}$	$c_1 \leftarrow g_1^{r}$; $c_2 \leftarrow g_2^{r}$	$K \leftarrow \mathsf{H}(c_1^{x_1} c_2^{x_2}, c_1^{\hat{x}_1} c_2^{\hat{x}_2})$
Pick $\mathsf{H} \leftarrow_R \mathcal{H}$	$K \leftarrow \mathsf{H}(X^r, \hat{X}^r)$	Return $\{m, \perp\} \leftarrow \mathsf{D}_K(\psi)$
$pk \leftarrow (X, \hat{X}, \mathsf{H})$	$\psi \leftarrow \mathsf{E}_K(m)$	
$sk \leftarrow (x_1, x_2, \hat{x}_1, \hat{x}_2)$	Return $C = (c_1, c_2, \psi)$	
Return (sk, pk)		

Fig. 3. Hybrid encryption scheme $\mathsf{HE}_2 = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ obtained by applying our randomness extraction framework to the HPS from Section 5.3

Relation to a scheme by Kurosawa and Desmedt. We remark that, interestingly, the scheme HE_2 is quite similar to the one by Kurosawa and Desmedt [17]. The only difference is that encryption in the latter defines the key as $K = X^{rt} \cdot \hat{X}^r \in \mathbb{G}$, where $t = \mathsf{T}(c_1, c_2)$ is the output of a (keyed) target collision-resistant hash function $\mathsf{T} : \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_p$.

5.4 Efficiency Considerations

In this section we compare the efficiency of $\mathsf{HE}_1/\mathsf{HE}_2$ and their explicit rejection variants $\mathsf{HE}_1^{\mathsf{er}}/\mathsf{HE}_2^{\mathsf{er}}$ with the reference scheme KD by Kurosawa and Desmedt [17] and its variants [11,14].

The drawback of HE_1 is that, in terms of concrete security, Theorem 9 requires the image $\{0,1\}^{\ell}$ of H to be sufficiently small, i.e., $\ell \leq \frac{1}{4} \lg p$. Consequently, for a symmetric cipher with $\ell = k = 80$ bits keys we are forced to use groups of order $\lg p \geq 4k = 320$ bits. For some specific groups such as elliptic curves this can be a drawback since there one typically works with groups of order $\lg p = 2k = 160$ bits. However, for other more traditional groups such as prime subgroups of

Table 1. Efficiency comparison for known CCA2-secure encryption schemes from the DDH assumption. All "symmetric" operations concerning the authenticated encryption scheme are ignored. The symbols "tcr" and "4wh" denote one application of a target collision-resistant hash function and 4-wise independent hash function, respectively.

Scheme	Assumption	Encryption	Decryption	Ciphertext	Key-size		Restriction
		#[multi/sequential,single]-exp		Size	Public	Secret	on $p = \|\mathbb{G}\|$
KD	DDH & TCR	$[1,2]$+tcr	$[1,0]$+tcr	$2\|\mathbb{G}\|+\|\psi\|$	$4\|\mathbb{G}\|+\|\mathsf{T}\|$	$4\|\mathbb{Z}_p\|$	$\lg p \geq 2\ell(k)$
$\mathsf{HE}_1^{\mathsf{er}}$	DDH	$[0,3]$+4wh	$[1,0]$+4wh	$2\|\mathbb{G}\|+\|\psi\|$	$3\|\mathbb{G}\|+\|\mathsf{H}\|$	$2\|\mathbb{Z}_p\|$	$\lg p \geq 4\ell(k)$
$\mathsf{HE}_2^{\mathsf{er}}$	DDH	$[0,4]$+4wh	$[1,0]$+4wh	$2\|\mathbb{G}\|+\|\psi\|$	$4\|\mathbb{G}\|+\|\mathsf{H}\|$	$4\|\mathbb{Z}_p\|$	$\lg p \geq 2\ell(k)$

$\mathbb{Z}_q^*$ one sometimes takes a subgroup of order already satisfying the requirement $\lg p \geq 4k$. The scheme HE_2 overcomes this restriction at the cost of an additional exponentiation in the encryption algorithm.

Table 1 summarizes the efficiency of the schemes KD [17], $\mathsf{HE}_1^{\mathsf{er}}$, and $\mathsf{HE}_2^{\mathsf{er}}$. (A comparison of the explicit rejection variants seems more meaningful.) It is clear that when groups of similar size are used, our new scheme $\mathsf{HE}_1^{\mathsf{er}}$ will be the most efficient. But, as detailed above, typically $\mathsf{HE}_1^{\mathsf{er}}$ will have to work in a larger (sub)group. Even when underlying operations such as multiplication and squaring remain the same, the increased exponent length will make this scheme noticeably slower than the other two options.

6 Instantiations from the Quadaratic Residuosity Assumption

Quadratic residuosity assumption. Let $b = b(k) : \mathbb{N} \to \mathbb{N}^{>0}$ be a function. Let $N = pq$ be an RSA modulus consisting of distinct safe primes of bit-length $b/2$, i.e., $p = 2P + 1$ and $q = 2Q + 1$ for two primes P, Q. Let $\mathbb{J}_N$ denote the (cyclic) subgroup of elements in $\mathbb{Z}_N^*$ with Jacobi symbol 1, and let $\mathbb{QR}_N$ denote the unique (cyclic) subgroup of $\mathbb{Z}_N^*$ of order PQ (so in particular $\mathbb{QR}_N \subset \mathbb{J}_N$) which is the group of all squares modulo N. We assume the existence of an RSA instance generator RSAgen that generates the above elements, together with a random generator $g \in \mathbb{QR}_N$. The quadratic residuosity (QR) assumption states that distinguishing a random element from $\mathbb{QR}_N$ from a random element from $\mathbb{J}_N$ is computationally infeasible.

The hash proof system. Define $\mathcal{C} = \mathbb{J}_N$ and $\mathcal{V} = \mathbb{QR}_N = \{g^r \ : \ r \in \mathbb{Z}_{PQ}\}$. The value $r \in \mathbb{Z}$ is a witness of $C \in \mathcal{V}$. (Note that it is possible to sample an almost uniform element from $\mathcal{V}$ together with a witness by first picking $r \in \mathbb{Z}_{\lfloor N/4 \rfloor}$ and defining $C = g^r$.) Define $\mathcal{SK} = \mathbb{Z}_{2PQ}^n$, $\mathcal{PK} = \mathbb{QR}_N^n$, and $\mathcal{K} = \mathbb{J}_N^n$. For $sk = (x_1, \ldots, x_n) \in \mathbb{Z}_{2PQ}^n$, define $\mu(sk) = (X_1, \ldots, X_n) = (g^{x_1}, \ldots, g^{x_n})$.

For $C \in \mathcal{C}$ define

$$\Lambda_{sk}(C) := (C^{x_1}, \ldots, C^{x_n}) \ .$$

This defines $\mathsf{Priv}(sk, C)$. Given $pk = \mu(sk)$, $C \in \mathcal{V}$ and a witness $r \in \mathbb{Z}_{PQ}$ such that $C = g^r$, public evaluation $\mathsf{Pub}(pk, C, r)$ computes $K = \Lambda_{sk}(C)$ as

$$K = (X_1^r, \ldots, X_n^r) .$$

This completes the description of HPS. Under the QR assumption, the subset membership problem is hard in HPS. (The statistical difference between the uniform distribution over $\mathbb{QR}_N$ and the proposed way of sampling above, is at most $2^{-b/2}$, causing only a small extra term between the QR advantage and the HPS membership advantage.)

Consider a pair (X_i, x_i), where x_i is from sk and X_i is from pk and note that X_i does not reveal whether $0 \le x_i < PQ$ or $PQ \le x_i < 2PQ$. Therefore, for $C \in \mathcal{C} \setminus \mathcal{V}$, given $pk = \mu(sk)$, each of the C^{x_i} contains exactly one bit of min entropy such that $H_\infty((C^{x_1}, \ldots, C^{x_n}) \mid pk) = n$. Therefore:

Lemma 12. *The hash proof system is n-entropic with collision probability $\delta = 2^{-n}$.*

The encryption scheme. Let $\mathsf{H} : \mathbb{J}_N^n \to \{0,1\}^k$ be a 4-wise independent hash function and let SE be a symmetric cipher with key-space $\{0,1\}^k$, i.e., we set $\ell(k) = k$. For the encryption scheme obtained by applying Corollary 7 (which is depicted in Fig. 4) we choose the parameter $n = n(k) = 4k + 1$ such that $k - (n-1)/2 = -k$ so we can bound ϵ_2 by $2^{-k} + 2^{-n}$ using Theorem 6.

Theorem 13. *Assume the QR assumption holds, let $\mathcal{H}$ be a family of 4-wise independent hash functions from $\mathbb{J}_N^{n(k)}$ to $\{0,1\}^k$ with $n(k) \ge 4k + 1$, and let SE be a symmetric encryption that is secure in the sense of $\mathsf{AE\text{-}OT}$. Then the encryption scheme from Fig. 4 is $\mathsf{IND\text{-}CCA2}$ secure. In particular,*

$$\mathsf{Adv}^{\mathrm{cca2}}_{\mathsf{PKE},t,Q}(k) \le 2^{-b/2} + \mathsf{Adv}^{\mathrm{qr}}_{\mathcal{GS},t}(k) + 2Q \cdot \mathsf{Adv}^{\mathrm{int\text{-}ot}}_{\mathsf{SE},t}(k) + \mathsf{Adv}^{\mathrm{ind\text{-}ot}}_{\mathsf{SE},t}(k) + Q2^{-k+1} .$$

The scheme has very compact ciphertexts but encryption/decryption are quite expensive since they require $n = 4k + 1$ exponentiations in $\mathbb{Z}_N^*$. (Note that decryption can be sped up considerably compared to encryption by using CRT and multi-exponentiation techniques.)

$\mathsf{Kg}(1^k)$	$\mathsf{Enc}(pk, m)$	$\mathsf{Dec}(sk, C)$
$(N, P, Q, g) \leftarrow_R \mathsf{RSAgen}(1^k)$	$r \leftarrow_R \mathbb{Z}_{\lfloor N/4 \rfloor}$	Parse C as (c, ψ)
For $i = 1$ to $n := 4k + 1$ do	$c \leftarrow g^r$	$K \leftarrow \mathsf{H}(c^{x_1}, \ldots, c^{x_n})$
$\quad x_i \leftarrow_R \mathbb{Z}_{2PQ}$; $X_i \leftarrow g^{x_i}$	$K \leftarrow \mathsf{H}(X_1^r, \ldots, X_n^r)$	Return
Pick $\mathsf{H} \leftarrow_R \mathcal{H}$	$\psi \leftarrow \mathsf{E}_K(m)$	$\quad \{m, \perp\} \leftarrow \mathsf{D}_K(\psi)$
$pk \leftarrow (N, g, (X_i), \mathsf{H})$; $sk \leftarrow ((x_i))$	Return $C = (c, \psi)$	
Return (sk, pk)		

Fig. 4. Hybrid encryption scheme $\mathsf{HE}_3 = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ obtained by applying our randomness extraction framework to the HPS from Section 6

Acknowledgements

We thank Ronald Cramer for interesting discussions. We are furthermore grateful to Victor Shoup for pointing out the scheme from Section 5.3. We thank Kenny Paterson, Steven Galbraith and James Birkett for useful feedback, prompting the comparison in Section 5.4.

References

1. Choi, S.G., Herranz, J., Hofheinz, D., Hwang, J.Y., Kiltz, E., Lee, D.H., Yung, M.: The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure. Information Processing Letters (to appear, 2009)
2. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
3. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
4. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
5. Damgård, I.B.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
6. Damgård, I.B., Goldreich, O., Okamoto, T., Wigderson, A.: Honest verifier vs dishonest verifier in public coin zero-knowledge proofs. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 325–338. Springer, Heidelberg (1995)
7. Desmedt, Y., Lipmaa, H., Phan, D.H.: Hybrid Damgård is CCA1-secure under the DDH assumption. In: CANS 2008. LNCS, vol. 5339, pp. 18–30. Springer, Heidelberg (2008)
8. Desmedt, Y., Phan, D.H.: A CCA secure hybrid Damgårds ElGamal encryption. In: ProvSec 2008. LNCS, vol. 5324, pp. 68–82. Springer, Heidelberg (2008)
9. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
10. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
11. Gennaro, R., Shoup, V.: A note on an encryption scheme of Kurosawa and Desmedt. Cryptology ePrint Archive, Report 2004/194 (2004), `http://eprint.iacr.org/`
12. Gjøsteen, K.: A new security proof for Damgård's ElGamal. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 150–158. Springer, Heidelberg (2006)
13. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
14. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
15. Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)

16. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. Cryptology ePrint Archive, Report 2008/304 (2008), http://eprint.iacr.org/
17. Kurosawa, K., Desmedt, Y.G.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
18. Lipmaa, H.: On CCA1-Security of Elgamal and Damgård cryptosystems. Cryptology ePrint Archive, Report 2008/234 (2008), http://eprint.iacr.org/
19. Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
20. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. ACM Press, New York (1990)
21. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
22. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
23. Shacham, H.: A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007), http://eprint.iacr.org/
24. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)
25. Tsiounis, Y., Yung, M.: On the security of ElGamal based encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 117–134. Springer, Heidelberg (1998)
26. Wu, J., Stinson, D.R.: On the security of the ElGamal encryption scheme and Dåmgard's variant. Cryptology ePrint Archive, Report 2008/200 (2008), http://eprint.iacr.org/

Erratum to: Advances in Cryptology – EUROCRYPT 2009

Antoine Joux

DGA and University of Versailles Saint-Quentin-en-Yvelines,
45, avenue des Etats-Unis, 78035 Versailles Cedex, France
antoine.joux@m4x.org

Erratum to:
A. Joux (Ed.)
Advances in Cryptology – EUROCRYPT 2009
DOI: 10.1007/978-3-642-01001-9

The book was inadvertently published with an incorrect name of the copyright holder. The name of the copyright holder for this book is: © Springer-Verlag Berlin Heidelberg. The book has been updated with the changes.

The updated original online version for this book can be found at
DOI: 10.1007/978-3-642-01001-9

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, p. E1, 2009.

Author Index